Radio Shack®
TRS-80
MODEL 100
PORTABLE
COMPUTER

TRS-80®

MODEL 100
PORTABLE COMPUTER

Dec 28,1982 Tue 14:00:37 (C)Microsoft
BASIC TEXT TELCOM ADDRSS
SCHEDL ADRS.DO NOTE.DO
Select: _ 29102 Bytes free
Radio Shack
TRS-80
MODEL 100 PORTABLE COMPUTER

CUSTOM MANUFACTURED FOR RADIO SHACK, A DIVISION OF TANDY CORPORATION

TERMS AND CONDITIONS OF SALE AND LICENSE OF RADIO SHACK COMPUTER EQUIPMENT AND SOFTWARE
PURCHASED FROM A RADIO SHACK COMPANY-OWNED COMPUTER CENTER, RETAIL STORE OR FROM A
RADIO SHACK FRANCHISEE OR DEALER AT ITS AUTHORIZED LOCATION

LIMITED WARRANTY

I. CUSTOMER OBLIGATIONS

A. CUSTOMER assumes full responsibility that this Radio Shack computer hardware purchased (the "Equipment"), and any copies of Radio Shack software included with the Equipment or licensed separately (the "Software") meets the specifications, capacity, capabilities, versatility, and other requirements of CUSTOMER.

B. CUSTOMER assumes full responsibility for the condition and effectiveness of the operating environment in which the Equipment and Software are to function, and for its installation.

II. RADIO SHACK LIMITED WARRANTIES AND CONDITIONS OF SALE

A. For a period of ninety (90) calendar days from the date of the Radio Shack sales document received upon purchase of the Equipment, RADIO SHACK warrants to the original CUSTOMER that the Equipment and the medium upon which the Software is stored is free from manufacturing defects. THIS WARRANTY IS ONLY APPLICABLE TO PURCHASES OF RADIO SHACK EQUIPMENT BY THE ORIGINAL CUSTOMER FROM RADIO SHACK COMPANY-OWNED COMPUTER CENTERS, RETAIL STORES AND FROM RADIO SHACK FRANCHISEES AND DEALERS AT ITS AUTHORIZED LOCATION. The warranty is void if the Equipment's case or cabinet has been opened, or if the Equipment or Software has been subjected to improper or abnormal use. If a manufacturing defect is discovered during the stated warranty period, the defective Equipment must be returned to a Radio Shack Computer Center, a Radio Shack retail store, participating Radio Shack franchisee or Radio Shack dealer for repair, along with a copy of the sales document or lease agreement. The original CUSTOMER'S sole and exclusive remedy in the event of a defect is limited to the correction of the defect by repair, replacement, or refund of the purchase price, at RADIO SHACK'S election and sole expense. RADIO SHACK has no obligation to replace or repair expendable items.

B. RADIO SHACK makes no warranty as to the design, capability, capacity, or suitability for use of the Software, except as provided in this paragraph. Software is licensed on an "AS IS" basis, without warranty. The original CUSTOMER'S exclusive remedy, in the event of a Software manufacturing defect, is its repair or replacement within thirty (30) calendar days of the date of the Radio Shack sales document received upon license of the Software. The defective Software shall be returned to a Radio Shack Computer Center, a Radio Shack retail store, participating Radio Shack franchisee or Radio Shack dealer along with the sales document.

C. Except as provided herein no employee, agent, franchisee, dealer or other person is authorized to give any warranties of any nature on behalf of RADIO SHACK.

D. Except as provided herein, **RADIO SHACK MAKES NO WARRANTIES, INCLUDING WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.**

E. Some states do not allow limitations on how long an implied warranty lasts, so the above limitation(s) may not apply to CUSTOMER.

III. LIMITATION OF LIABILITY

A. EXCEPT AS PROVIDED HEREIN, RADIO SHACK SHALL HAVE NO LIABILITY OR RESPONSIBILITY TO CUSTOMER OR ANY OTHER PERSON OR ENTITY WITH RESPECT TO ANY LIABILITY, LOSS OR DAMAGE CAUSED OR ALLEGED TO BE CAUSED DIRECTLY OR INDIRECTLY BY "EQUIPMENT" OR "SOFTWARE" SOLD, LEASED, LICENSED OR FURNISHED BY RADIO SHACK, INCLUDING, BUT NOT LIMITED TO, ANY INTERRUPTION OF SERVICE, LOSS OF BUSINESS OR ANTICIPATORY PROFITS OR CONSEQUENTIAL DAMAGES RESULTING FROM THE USE OR OPERATION OF THE "EQUIPMENT" OR "SOFTWARE". IN NO EVENT SHALL RADIO SHACK BE LIABLE FOR LOSS OF PROFITS, OR ANY INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY BREACH OF THIS WARRANTY OR IN ANY MANNER ARISING OUT OF OR CONNECTED WITH THE SALE, LEASE, LICENSE, USE OR ANTICIPATED USE OF THE "EQUIPMENT" OR "SOFTWARE".

NOTWITHSTANDING THE ABOVE LIMITATIONS AND WARRANTIES, RADIO SHACK'S LIABILITY HEREUNDER FOR DAMAGES INCURRED BY CUSTOMER OR OTHERS SHALL NOT EXCEED THE AMOUNT PAID BY CUSTOMER FOR THE PARTICULAR "EQUIPMENT" OR "SOFTWARE" INVOLVED.

B. RADIO SHACK shall not be liable for any damages caused by delay in delivering or furnishing Equipment and/or Software.

C. No action arising out of any claimed breach of this Warranty or transactions under this Warranty may be brought more than two (2) years after the cause of action has accrued or more than four (4) years after the date of the Radio Shack sales document for the Equipment or Software, whichever first occurs.

D. Some states do not allow the limitation or exclusion of incidental or consequential damages, so the above limitation(s) or exclusion(s) may not apply to CUSTOMER.

IV. RADIO SHACK SOFTWARE LICENSE

RADIO SHACK grants to CUSTOMER a non-exclusive, paid-up license to use the RADIO SHACK Software on **one** computer, subject to the following provisions:

A. Except as otherwise provided in this Software License, applicable copyright laws shall apply to the Software.

B. Title to the medium on which the Software is recorded (cassette and/or diskette) or stored (ROM) is transferred to CUSTOMER, but not title to the Software.

C. CUSTOMER may use Software on one host computer and access that Software through one or more terminals if the Software permits this function.

D. CUSTOMER shall not use, make, manufacture, or reproduce copies of Software except for use on **one** computer and as is specifically provided in this Software License. Customer is expressly prohibited from disassembling the Software.

E. CUSTOMER is permitted to make additional copies of the Software **only** for backup or archival purposes or if additional copies are required in the operation of **one** computer with the Software, but only to the extent the Software allows a backup copy to be made. However, for TRSDOS Software, CUSTOMER is permitted to make a limited number of additional copies for CUSTOMER'S own use.

F. CUSTOMER may resell or distribute unmodified copies of the Software provided CUSTOMER has purchased one copy of the Software for each one sold or distributed. The provisions of this Software License shall also be applicable to third parties receiving copies of the Software from CUSTOMER.

G. All copyright notices shall be retained on all copies of the Software.

V. APPLICABILITY OF WARRANTY

A. The terms and conditions of this Warranty are applicable as between RADIO SHACK and CUSTOMER to either a sale of the Equipment and/or Software License to CUSTOMER or to a transaction whereby RADIO SHACK sells or conveys such Equipment to a third party for lease to CUSTOMER.

B. The limitations of liability and Warranty provisions herein shall inure to the benefit of RADIO SHACK, the author, owner and/or licensor of the Software and any manufacturer of the Equipment sold by RADIO SHACK.

VI. STATE LAW RIGHTS

The warranties granted herein give the **original** CUSTOMER specific legal rights, and the **original** CUSTOMER may have other rights which vary from state to state.

The FCC Wants You to Know . . .

This equipment generates and uses radio frequency energy. If not installed and used properly, that is, in strict accordance with the manufacturer's instructions, it may cause interference to radio and television reception.

It has been type tested and found to comply with the limits for a Class B computing device in accordance with the specifications in Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference in a residential installation. However, there is no guarantee that interference will not occur in a particular installation.

If this equipment does cause interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient the receiving antenna
- Relocate the computer with respect to the receiver
- Move the computer away from the receiver
- Plug the computer into a different outlet so that computer and receiver are on different branch circuits.

If necessary, you should consult the dealer or an experienced radio/television technician for additional suggestions. You may find the following booklet prepared by the Federal Communications Commission helpful: *How to Identify and Resolve Radio-TV Interference Problems.*

This booklet is available from the US Government Printing Office, Washington, DC 20402, Stock No. 004-000-00345-4.

Warning

This equipment has been certified to comply with the limits for a Class B computing device, pursuant to Subpart J of Part 15 of FCC Rules. Only peripherals (computer input/output devices, terminals, printers, etc.) certified to comply with the Class B limits may be attached to this computer. Operation with non-certified peripherals is likely to result in interference to radio and TV reception.

***TRS-80® Model 100 Owner's Manual:* ©1983 Tandy Corporation, Fort Worth, Texas 76102 U.S.A. All Rights Reserved.**

Reproduction or use, without express written permission from Tandy Corporation, of any portion of this manual is prohibited. While reasonable efforts have been taken in the preparation of this manual to assure its accuracy, Tandy Corporation assumes no liability resulting from any errors or omissions in this manual, or from the use of the information obtained herein.

TRS-80® Model 100 Software: ©1983 Microsoft Incorporated. All Rights Reserved.

The software in the Model 100 microcomputer is retained in a read-only memory (ROM) format. All portions of this software, whether in the ROM format or other source code form format, and the ROM circuitry, are copyrighted and are the proprietary and trade secret information of Tandy Corporation and Microsoft. Use, reproduction or publication of any portion of this material without the prior written authorization by Tandy Corporation is strictly prohibited.

Contents

Introduction 1

PART I / GETTING STARTED . . .

1 / Description of the Model 100 5
2 / Getting Power to the Computer 9
    Battery Installation 9
    Connecting to a Power Source 11
3 / Turning the Power On and Off 13
    Power-On 13
    Power-Off 13
4 / The Main Menu Screen 15
    Adjusting the Display Screen 16
    Selecting a Menu Item 16
5 / Setting the Date, Time, and Day 17
6 / Quick Instructions on Using the Applications Programs 19

PART II / THE MAIN MENU OPTIONS

7 / Main Menu Overview 39
8 / Text Preparation (TEXT) 43
9 / Scheduling Appointments (SCHEDL) 61
10 / Recording Addresses (ADDRSS) 69
11 / Computer-to-Computer Communication (TELCOM) 75

PART III / MODEL 100 BASIC

12 / BASIC Overview 99
13 / Data Types 103
14 / Control Commands 115
15 / Input/Output 121
16 / BASIC Keywords 127

PART IV / APPENDICES

Appendix A / Connecting the Model 100 to Optional Equipment 189
- Cassette Recorder 189
- Printer 191
- Another TRS-80 Computer 193

Appendix B / Notes on Power On/Off 195
- Warm vs. Cold Power-On/Off 195
- Changing the Auto-Power Off Settings 195

Appendix C / Sample Sessions 197

Appendix D / Technical Information 205
- Specifications 205
- Video Display Worksheet 210
- Character Code Tables 211
- BASIC Error Codes 217
- Derived Functions 218

Appendix E / Troubleshooting and Maintenance 219

Index 221

Introduction

Congratulations for selecting the TRS-80® Model 100 Portable Computer. The Model 100, which fits easily into a regular sized briefcase, has many special features and functions that make it the perfect portable computer for home or office — *or anywhere in between!*

Whether you're a computer professional or novice, you'll find the Model 100 simple to operate. Its special features include five ''built-in'' Application Programs:

- **TEXT** for text and word processing preparation.
- **TELCOM** for communication with other computers.
- **SCHEDL** to keep track of appointments and other schedules.
- **ADDRSS** to maintain addresses and phone numbers.
- **BASIC** that lets you write your own programs easily and quickly.

The Model 100 offers the convenience of battery operation for portability or AC power for home and office use.

Furthermore, a full-size keyboard provides eight programmable Function Keys, four Command Keys, four Cursor Movement Keys, and a 10-Key Numeric Pad.

The Model 100 also has its own built-in modem (for computer-to-computer communication over the telephone) as well as an RS-232C Interface, the most universal standard for linking computer equipment.

And when you have the Model 100 connected to a phone, you can also use the Computer as an automatic dialer that stores and dials hundreds of telephone numbers!

The Model 100 can be used with the following optional equipment:

- Cassette Recorder for program or data storage on cassette tapes.
- Parallel Printers (daisy wheel or dot-matrix) for printed copies of documents, programs, or data.
- Bar Code Reader for product marking-code identification (UPC, CODABAR, etc.). (Optional/extra software required.)

and more!

The Model 100 is available with 8K (26-3801) and 24K (26-3802) of Random Access Memory. You can expand this with the Model 100 RAM Upgrade Kit (26-3816) that provides an additional 8K of memory. (Installation required by a qualified Radio Shack service technician.)

About this manual . . .

For your convenience, we've divided this manual into four parts:

Part I describes the Model 100 and should get you started using your new computer right away. This section won't go into a lot of detail, but by the time you finish it, you'll be able to use your Model 100 for a variety of applications — including text preparation, appointment scheduling, and communicating with an information service.

Part II ''teaches'' you how to use the Application Programs introduced in Part I by explaining in detail and providing examples.

Part III describes Model 100 BASIC, the built-in programming language that allows you to write your own ''customized'' programs. This section *won't* try to teach you programming in general. It simply tells how *this* Computer uses BASIC. If you want to learn more about programming, we recommend ***Getting Started with TRS-80® BASIC*** (Radio Shack Catalog Number 26-2107).

Part IV, the Appendices, contains information about connecting and using the Model 100 with various optional equipment. It also provides technical information that you may need when using the Model 100.

We suggest you become familiar with this manual (especially Part I!) and the Model 100 before you begin using the Computer. After that, the ***Model 100 Quick Reference Guide*** (included with this package) should keep you up-and-running!

So take a few minutes now and get to know your new Portable Computer — and remember that a little extra time spent now may save you hours later on.

PART I / GETTING STARTED WITH THE MODEL 100

Since we know you'll want to get started with your TRS-80® Model 100 Portable Computer as quickly as possible, everything you'll need to know to use your Model 100 will be discussed in this section.

We'll start with a description of the Model 100. Read Chapter 1 carefully since many of the Computer's special features (including the keyboard) are unique to the Model 100.

Chapters 2 and 3 will show you how to load batteries into the Computer, connect it to an AC power source, and turn the power on and off.

Chapters 4 and 5 describe what you see on the Display when you turn the power on, and what you should do then.

Finally, Chapter 6 provides quick instructions on using the Model 100's built-in Application Programs.

What Chapter 6 won't do is give you an in-depth description of Model 100 Application Programs, BASIC Interpreter, and operating procedures — especially in regard to connection and operation of optional equipment (such as printers and cassette recorders). For a detailed discussion of these topics, see the appropriate sections later in the manual.

1 / Description of the Model 100

Open the package and take out the Model 100. Do not throw away the packing material or the box. They may be useful if you ever need to send the Computer through the mail.

The Model 100 package includes:

- A Model 100 Portable Computer
- This owner's manual
- A Quick Reference Guide

The Model 100 features a Liquid Crystal Display (LCD), a full-size, typewriter-style keyboard, and connectors that allow a variety of optional equipment to be attached to the Computer.

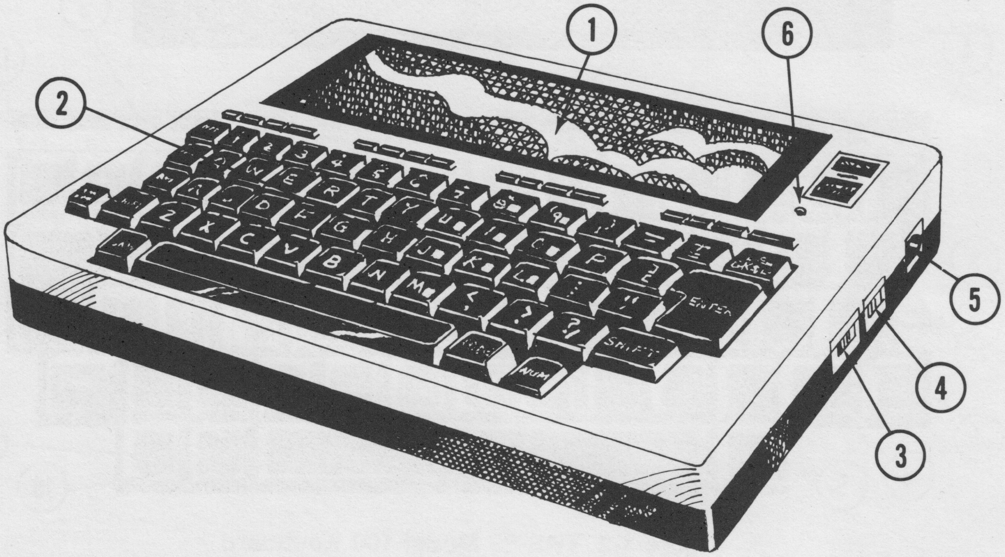


Figure 1-1. Model 100 (Front View)

① **LCD Screen** The Model 100 Display has eight lines that allow 40 characters on each line. The Screen can be adjusted for optimum viewability.

② **Keyboard** For most applications, the Keyboard can be used exactly like a standard typewriter. However, the Model 100 does have a few special keys (see Figure 1-2 for more details).

③ **Power ON/OFF Switch** Move this switch towards the *front* to turn the power ON. To conserve the batteries, the Model 100 automatically turns the power off if you do not use it for 10 minutes. (For details, see "**Power-Off**" in Chapter 3). When an automatic power-off occurs, the switch will still be in the ON position even though the power is OFF. To turn the power ON, move the switch to the OFF position, then back ON.

④ **Display Adjustment Dial** Adjust this dial for optimum viewability.

⑤ **External Power Adapter Connector** Connect the appropriate end of the Radio Shack AC Power Supply 26-3804 (optional/extra) to this connector. Use only this power supply! Connect the other end of the power supply to a 120 VAC wall-outlet or approved power strip.

⑥ **Low Battery Indicator** Before the Model 100's operational batteries become exhausted, this Indicator will illuminate. When it does, you have about 20 minutes of power remaining. You should replace the batteries as quickly as possible when the Indicator lights up.

The Model 100 Keyboard is capable of producing upper- and lowercase alphabetical characters as well as special and graphic characters. If held down, each key on the keyboard will continuously generate characters ("auto-repeat").

In addition, the keyboard contains:

- Eight Programmable Function Keys (F1 - F8)
- Four Command Keys (PRINT, LABEL, PAUSE, PASTE)
- Four Cursor Movement Keys (←, →, ↑, ↓)

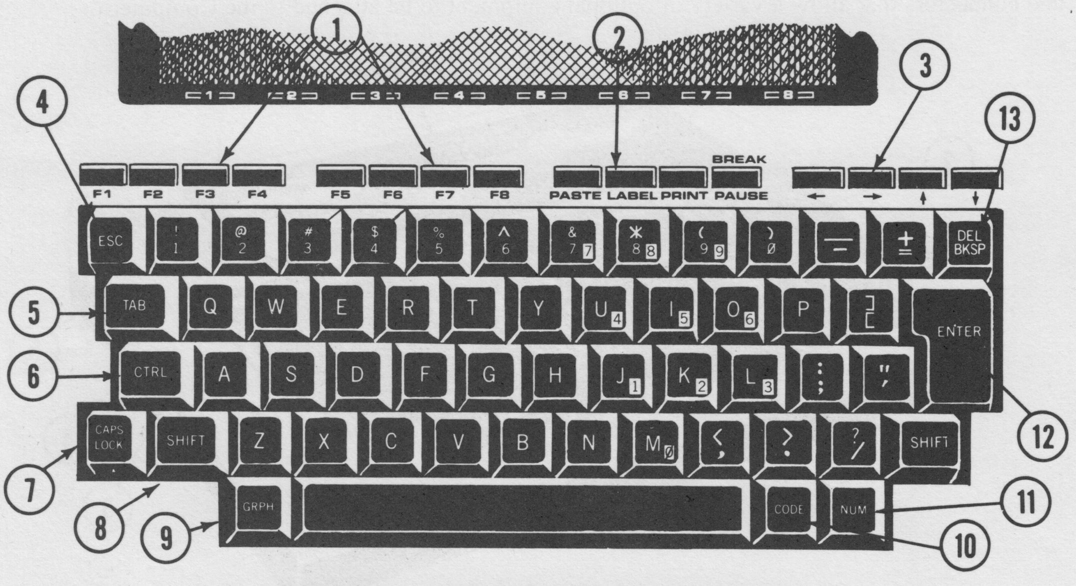


Figure 1-2. TRS-80 Model 100 Keyboard

① **Programmable Function Keys** Keys F1 through F8 have predefined functions in each of the five Application Programs. In each Application Program, the "current" definition of each key will be displayed above the appropriate Function Key. In BASIC, these keys may be programmed for "customized" operations.

② **Command Keys** The Model 100 keyboard has four predefined keys which perform specific operations. These operations are the same in all of the Application Programs.

- PASTE allows you to insert or move text that you previously defined.
- LABEL displays the definitions of the keys F1 through F8 in the current Application Program.
- PRINT lets you print out on a printer whatever is currently displayed on the Display. (In BASIC or TEXT, SHIFT PRINT will print the entire "file.")
- PAUSE causes program execution to halt temporarily. BREAK (SHIFT PAUSE) "breaks" program execution completely.

③ **Cursor Movement Keys** These four "arrow" keys move the Cursor on the Display. They perform the same operations in all of the Application Programs.

④ ESC This key is the "escape" key.

⑤ TAB This key advances the Cursor to the next predetermined ''tab'' setting (eight spaces at a time).

⑥ CTRL This is the ''control'' key. It can be used in conjunction with alpha keys to send Control Codes. In TEXT, it also allows a variety of Cursor movement options.

⑦ CAPS LOCK When you press this key, you will ''lock-in'' uppercase letters. Press again to ''unlock'' capital letters, allowing the option of upper- or lowercase letters. CAPS LOCK affects only the alpha-keys. If you need to type the characters at the top of the other keys, press SHIFT.

⑧ SHIFT This key can be used the same as with a standard typewriter. SHIFT, pressed in conjunction with any other key, will produce uppercase letters or a key's alternate definition.

⑨ GRPH When this key is pressed in conjunction with another key, 39 ''special'' Graphics characters become available. Pressing SHIFT GRPH followed by another key provides 34 additional ''block'' Graphics characters.

⑩ CODE Pressing this key provides 32 additional special characters. Pressing SHIFT CODE followed by another key enables more special characters.

⑪ NUM When this key is pressed, the keys with numbers in the lower-right corner can be used as a 10-key Numeric Pad.

⑫ ENTER This key is used like the carriage return on a conventional typewriter. In most applications, press ENTER at the end of each statement.

⑬ BKSP Pressing this key will erase the character which is to the left of the Cursor. Pressing DEL (SHIFT BKSP) will erase the character the Cursor is directly on top of.

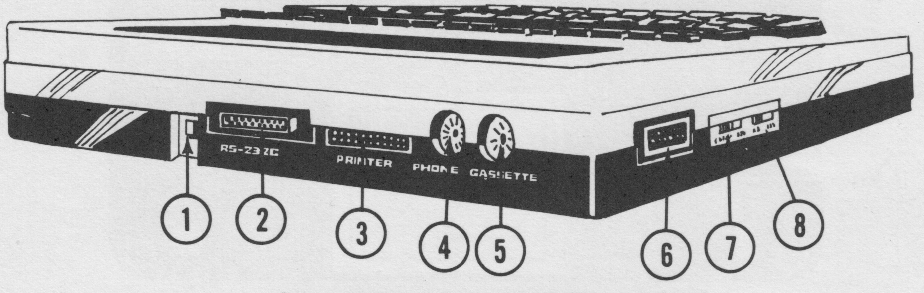


Figure 1-3. Model 100 (Rear View)

① **RESET Button** If the Model 100 ever ''locks-up'' (i.e., the Display will ''freeze'' and all keys seem to be inoperative), press this button to return to the Main Menu (''start-up'') Screen. It's highly unlikely that the Model 100 will ever lock-up when you're using the built-in Application Programs. However, this situation may occur with customized programs.

② **RS-232C Connector** Attach a DB-25 cable (such as Radio Shack Catalog Number 26-1408) to this connector when you need to receive or transmit serial information. When communicating directly with another TRS-80 computer, a ***Null Modem Adapter*** (26-1496) is required. An ***8" Cable Extender*** (26-1497) may also be required.

③ **PRINTER Connector** For hard-copy printouts of information, attach any Radio Shack parallel printer to this connector using an optional/extra ***Model 100 Parallel Printer Cable*** (26-1409).

④ **Direct Connect Modem (PHONE) Connector** When communicating with another computer via the Model 100's built-in modem, connect the round end of the optional/extra ***Model 100 Direct Connect Modem Cable*** (26-1410) to this connector. Be sure to set the Modem Switch (on the left side of the Model 100) to ***ANS***wer or ***ORIG***inate — depending on whether you are calling the other computer (''ORIGinating the call'') or it is calling you.

⑤ **CASSETTE Recorder Connector** To save or load information on a cassette tape, connect the cassette recorder here. (Optional/extra cassette recorder required. We suggest the ***CCR-81 Computer Recorder*** (26-1208) and C-20 leaderless computer cassette tapes (26-301).)

⑥ **Bar Code Wand Connector** Attach the optional/extra Bar Code Wand to this connector. Note that special Bar Code Reader software is required.

⑦ **Direct Connect/Acoustic Coupler Modem Selector** If you are communicating with another computer over the phone lines via the built-in Direct Connect Modem, set this switch to **DIR**ect Connect. If you are using the optional/extra ***Model 100 Acoustic Coupler*** (26-3805), set this selector to **ACOU**stic.

⑧ **ANSwer/ORIGinate Selector** If you are ''originating'' a phone call to another computer, set this switch to **ORIG**. If another computer is calling your Model 100, set to **ANS**.

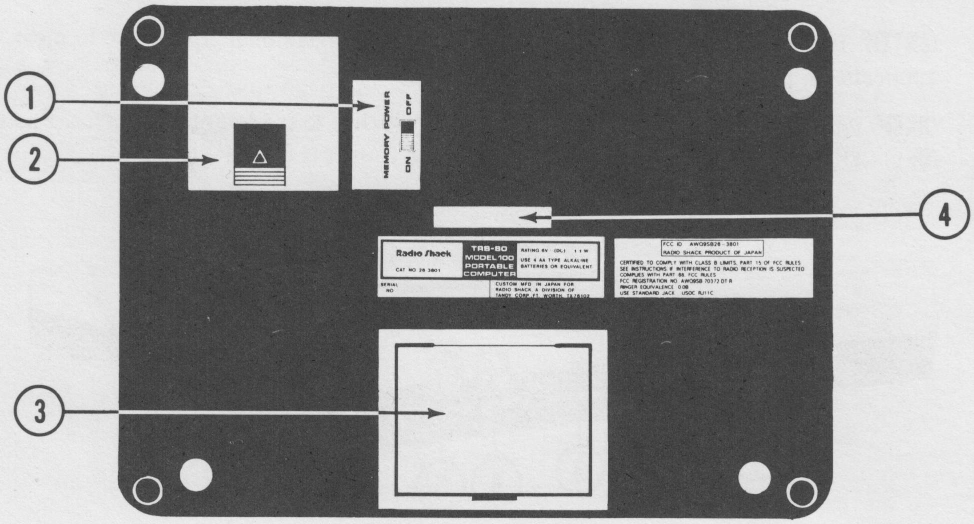

1
2
3
4
MEMORY POWER
ON
OFF
Radio Shack
CAT NO 26-3801
TRS-80
MODEL 100
PORTABLE
COMPUTER
RATING 6V (DC) 1.1 W
USE 4 AA TYPE ALKALINE
BATTERIES OR EQUIVALENT
SERIAL NO
CUSTOM MFD IN JAPAN FOR
RADIO SHACK A DIVISION OF
TANDY CORP. FT. WORTH, TX 76102
FCC ID AWO9SB26-3801
RADIO SHACK PRODUCT OF JAPAN
CERTIFIED TO COMPLY WITH CLASS B LIMITS, PART 15 OF FCC RULES
SEE INSTRUCTIONS IF INTERFERENCE TO RADIO RECEPTION IS SUSPECTED
COMPLIES WITH PART 68, FCC RULES
FCC REGISTRATION NO. AWO9SB 70372 DT R
RINGER EQUIVALENCE 0.0B
USE STANDARD JACK USOC RJ11C

Figure 1-4. Model 100 (Bottom View)

① **Memory Power Switch** Set this switch to ON after installing the batteries. The Model 100 *will not* operate if this switch is off even when the Power Switch is set to on.

Important Note! When you set this switch to OFF, *all* information stored in the Computer's memory is erased.

② **Battery Compartment** When not connected to an AC power source, the Model 100 gets its power from four AA size operational batteries that must be installed in this compartment.

③ **ROM Module Expansion Compartment** An optional/extra ROM Module Cartridges can be inserted into this compartment to further expand your Model 100's capabilities.

④ **ID Tag** Be sure to label the ID tag included with this package.

2 / Getting Power to the Computer

The TRS-80® Model 100 provides the convenience of battery operation as well as standard 120VAC power for extended use.

Battery Installation

The Model 100 uses four, size AA Alkaline operational batteries. We suggest you use Radio Shack Catalog Number 23-552.

The Model 100 also contains a Ni-Cad battery that is automatically re-charged whenever you use the Model 100. This battery provides the power to store programs and data when the Model 100 is turned off. How long the built-in Ni-Cad battery remains charged depends on how much RAM (memory) your Model 100 has. An 8K unit will retain all information for about 30 days after last power-on; a 32K unit will retain all information for about eight days.

How long the four operational batteries last depends on how often you use the Computer. For instance:

Computer Use Per Day	Estimated Battery Life
1 hour	20 days
4 hours	5 days

Table 2-1

To install the four operational batteries:

1. Remove the Battery Cover (on the bottom of the Computer case) by sliding it in the direction indicated by the arrow.
2. Position the batteries as shown in Figure 2-1.

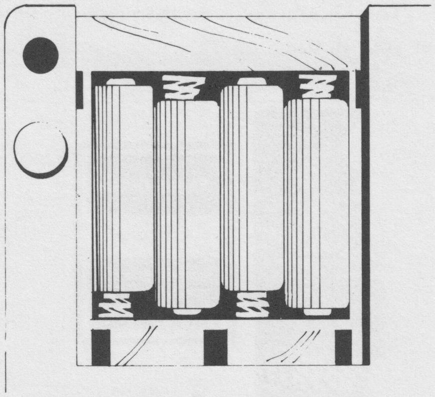


Figure 2-1. Battery Polarity

3. Replace the Battery Cover.

The Memory Power Switch

When you first take the Model 100 out of the box, the Memory Power Switch will be set to OFF.

After installing the four operational batteries, set the Memory Power Switch (next to the Battery Compartment) to ON. The Model 100 will not operate if this switch is set to OFF even when the Power Switch is set to ON.

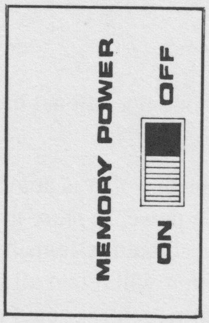


Figure 2-2. Memory Power Switch

Important! You *should not* set the Memory Power Switch to the OFF position when simply replacing the four operational batteries. The only time you really need to set this Switch to OFF is when you aren't using the Computer for extended periods of time. Be aware, however, that setting the Memory Power Switch to OFF erases all information (programs and data) currently stored in the Computer. If you intend on storing the Computer for an extended time and setting the Memory Power Switch to OFF, save all of your programs and data on cassette tape first.

The Low Battery Indicator

The Low Battery Indicator light, next to the Display, will indicate if the four AA operational batteries are low on power. When this Indicator lights up, you should immediately replace the AA batteries with fresh ones since you can only use the Computer for about 20 minutes more.

The Ni-Cad battery will keep all programs and data intact for eight to 30 days — depending on how much RAM is in the unit. (This should be more than enough time to finish your plane flight and get a new set of batteries at your destination.)

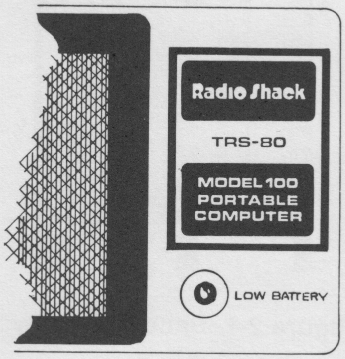


Figure 2-3. Low Battery Indicator

Connecting the Model 100 to an AC Power Source

By connecting the ***Model 100 Power Adapter*** (26-3804, optional/extra) to the Model 100, the Computer can be operated when the Adapter is plugged into a 120 VAC wall outlet.

Before connecting the Adapter to the Computer, be sure to turn the Computer and all peripherals OFF.

To connect the Model 100 to an AC wall outlet:

1. Connect the Adapter to an AC outlet.
2. Connect the Adapter to the **DC6V** Connector (located on the right side of the Computer). See Figure 2-4.

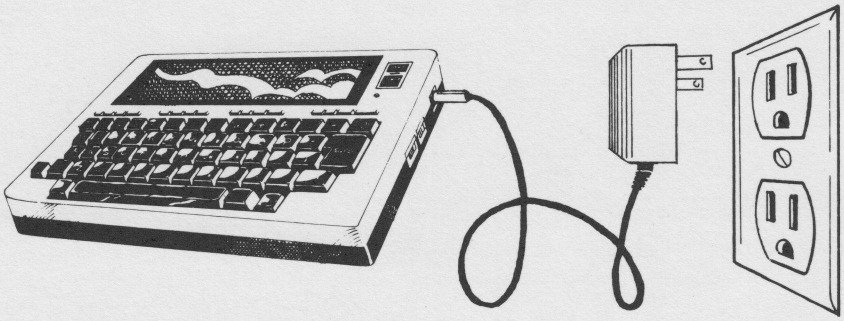


Figure 2-4. Connecting the Model 100 to a Wall-Outlet

3 / Turning the Power On and Off

Before setting the Power Switch to ON, be sure the Memory Power Switch is set to ON or the Computer will not operate even when the Power Switch is positioned to ON.

Turning the Power ON

When turning the Model 100's power on, follow this sequence:

1. Turn the Computer ON.
2. Turn all optional equipment (such as a printer) ON.

Turning the Power OFF

To turn the power OFF, simply move the Power Switch to the OFF position.

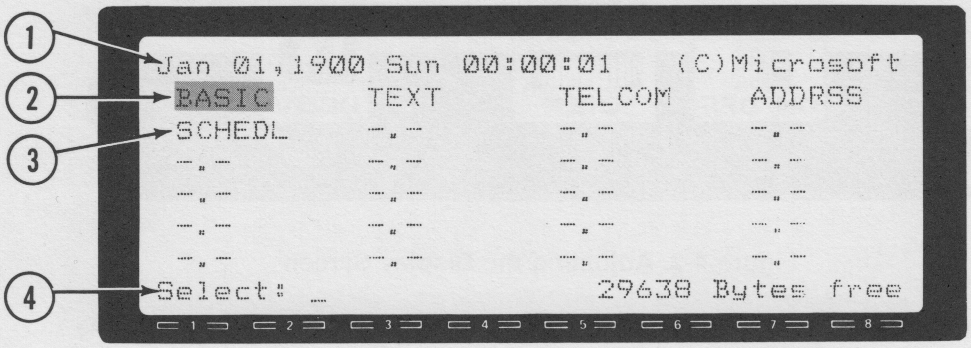
To prolong battery life, the Model 100 will turn itself off if you do not press any keyboard keys within a specific length of time. When the Model 100 is delivered, this time limit is set at approximately 10 minutes; however, you can change this time. See the **POWER** Command in Part III of this manual for details.

To protect the Model 100 and the information it stores, be sure to follow this procedure when turning the power OFF:

1. Turn all optional equipment (such as a printer) OFF.
2. Turn the Computer OFF.

4 / The Menu Screen

When you turn the Model 100's power on for the first time, the Display will look like Figure 4-1.



```
Jan 01,1900 Sun 00:00:01     (C)Microsoft
BASIC       TEXT        TELCOM      ADDRSS
SCHEDL      -.-         -.-         -.-
-.-         -.-         -.-         -.-
-.-         -.-         -.-         -.-
-.-         -.-         -.-         -.-
-.-         -.-         -.-         -.-
Select: _                   29638 Bytes free
```

Figure 4-1. Display Screen On Initial Power-Up

① The first line on the display indicates the date, day and time. On initial power-up, this line will be:

```
Jan 01, 1900  Sun  00:00:00
```

② The second line (and part of the third line) will list the names of the built-in Application Programs:

- BASIC
- TEXT (Text Preparation)
- TELCOM (Telecommunication)
- ADDRSS (Address Organzier)
- SCHEDL (Scheduler Organizer)

③ The remaining area, up to the seventh line, is reserved for the names of other programs or text documents you will create when using the Computer.

④ The bottom line on the Display allows you to select a text document (i.e., memos, daily schedules, addresses, etc.) or program (`SELECT:_`). It also displays the amount of the Computer's memory that is free for your use (*xxxx* Bytes Free). On initial power-up, a 32K unit will display 29638; a 24K unit, 21446; a 16K unit, 13254; and an 8K unit, 5062.

The Cursor

Notice the word BASIC (in the upper-left corner) is shaded. This shadow is the "Cursor" and can be moved to the name of any of the other Application Programs or text documents and programs you create that are listed on the Display.

To move the Cursor, press one of the Cursor Movement Keys (◄, ►, ▲, or ▼) or SPACE KEY.

Adjusting the Display Screen

Depending on the angle at which you're viewing the Display, you may need to adjust the Display Angle Control (labeled **DISP** and located next to the Power Switch) for optimum viewability.

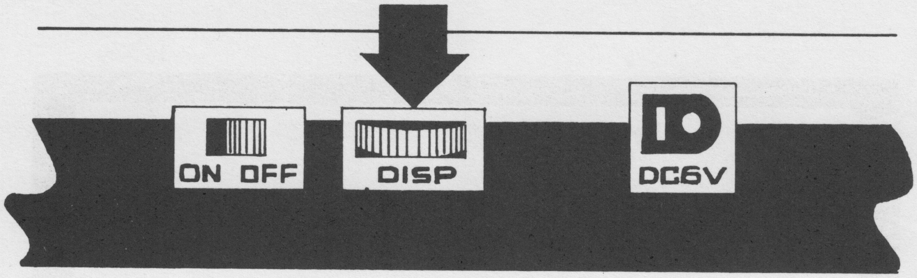


Figure 4-2. Adjusting the Display Screen

This adjustment allows you to view the Display from different positions without moving the Computer. Each time you change the position or angle of the Computer, you'll probably have to re-adjust the Display.

Selecting a Main Menu Option

When you power-up the Model 100, the Main Menu shows you the names of all ''files'' in the Computer.

Think of these Computer files as ordinary file folders that contain either programs or text documents. The Model 100 comes with five built-in files containing the Application Programs — BASIC, TEXT, TELCOM, ADDRSS, and SCHEDL.

You may create new text files and programs. When you do so, the names you assign these files and programs (''file names'') will also appear on the Main Menu.

You can access a file directly from the Main Menu.

To select a Main Menu Option:

1. Be sure the Cursor is positioned over the name of the Menu Option you want to select. Press the Cursor Movement Keys until it does.
2. When the Cursor is on top of the name you wish to use, press **ENTER**.

For instance, to load the Text Program, press → to position the Cursor so the shadow overlays the name `TEXT`. Then press **ENTER**.

5 / Setting the Time, Date, and Day

The first thing you should do when the Menu Screen appears is set the date, day, and time to the current values.

It's important to note that the Model 100 clock is a 24-hour or ''military'' clock. That is, 1:00 p.m. will be expressed as 13:00, 2:00 p.m. as 14:00 and so on.

Which value you set first is up to you. You can also reset any of the three values (date, day, or time) independently of the other two.

To enter these values, you'll have to use the BASIC Application Program. Values entered can be in either upper- or lowercase letters.

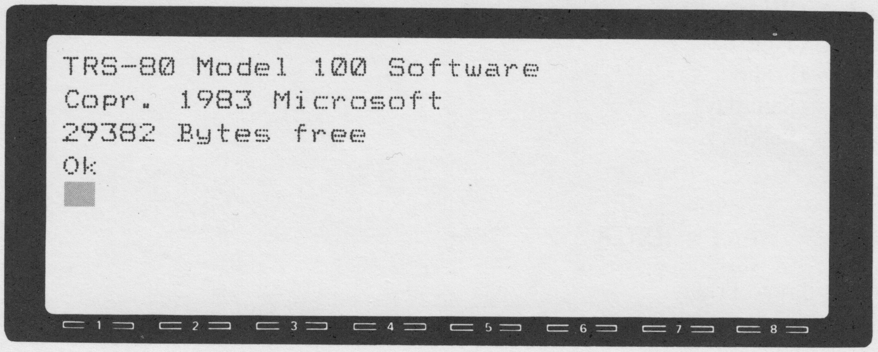
Important Note! If what you type is displayed as numbers instead of letters, be sure you haven't inadvertentlypressed the (NUM) (number) key down. When you take the Model 100 out of the box or out of it's carrying case, (NUM) or (CAPS) will probably be ''down.'' Press to release; then begin normal keyboard operation.

To set the time, day, or date . . .

1. First load BASIC.

   Be sure the Cursor is positioned over the word BASIC (press ▶ until it does).

2. When the Cursor is on top of `BASIC`, press (ENTER). You will then be in BASIC and the Screen will look like this:



```
TRS-80 Model 100 Software
Copr. 1983 Microsoft
29382 Bytes free
Ok
█
```

   Note that the blinking, BASIC Cursor (located below the word OK) is smaller than the Main Menu Cursor.

3. Then follow the procedures described in the rest of this chapter.

Note: If you have an 8K, 16K, or 24K Model 100, the number of bytes free will be different than the above Screen.

To set the time, type:

TIME$ = *"hour:minute:second"* and press ENTER

where *hour* is a two-digit number between 00 and 23,
minute is a two-digit number between 00 and 59, and
second is a two-digit number between 00 and 59. For instance:

```
TIME$ = "13:30:32" ENTER
```

would set the time at 1:30:32 p.m.

To set the date, type:

DATE$ = *"month/day/year"* and press ENTER

where *month* is a two-digit number between 01 and 12,
day is a two-digit number 01 and 31, and *year* is a two-digit number between 00 and 99. For instance:

```
DAY$ = "04/13/82" ENTER
```

would set the date for April 13, 1982.

To set the day, type:

DAY$ = *"day"* and press ENTER

where *day* is one of the following three-letter abbreviations:

Mon	(Monday)
Tue	(Tuesday)
Wed	(Wednesday)
Thu	(Thursday)
Fri	(Friday)
Sat	(Saturday)
Sun	(Sunday)

For instance:

```
DAY$ = "FRI" ENTER
```

would set the day to Friday.

Now the time, date, and day have been entered. To check that they're entered correctly, type:

```
PRINT TIME$ ENTER
```

and the time you entered will be displayed.

```
PRINT DATE$ ENTER
```

and the date you entered will be displayed.

```
PRINT DAY$ ENTER
```

and the day you entered will be displayed.

At this point, return to the Main Menu by pressing the Function Key F8. The date, day, and time you entered while in BASIC will be displayed on the first line.

6 / Quick Instructions for Using the Applications Programs

This section will briefly describe how to access and use the built-in Application Programs. For in-depth details and examples on using these programs, see the appropriate sections in Part II and Part III of this manual.

Using the Text Preparation Program (TEXT)

TEXT enables you to prepare text for word processing as well as to create files for SCHEDL, ADDRSS, and TELCOM.

There are a variety of ways to perform Text operations. For now, however, we'll only show you one way — the easiest way!

Using the Function Keys in TEXT

The Function Keys F1 - F8 have unique definitions when you're using the TEXT program.

To see these unique definitions, press the LABEL Command Key. The bottom line of the Display will look like Figure 6-1.

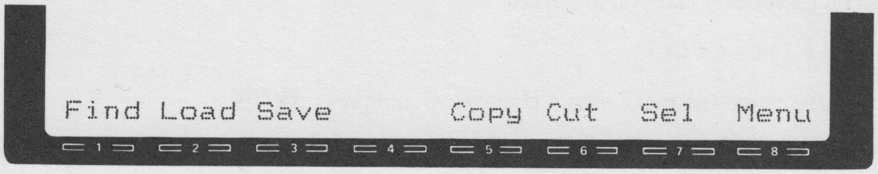


Figure 6-1. TEXT Function Key Definitions

Find Pressing F1 allows you to "find" a specified item. Press F1 and type in the "string" (a sequence of any characters — letters or numbers) you want to look for. The Cursor will move to the first occurrence in the current file of what you typed.

Load Pressing F2 allows you to get information from a device (such as a cassette recorder) into the Computer.

Save Pressing F3 lets you store information (onto a cassette recorder or other device, for instance).

F4 is not used by the Text Program.

Copy Press F5 to duplicate ("copy") text that has been defined with the "select" (F7) and Cursor Movement Keys.

Cut Press F6 to delete ("cut") text that has been defined with the "select" (F7) and Cursor Movement Keys.

Sel This is the "select" key. Press F7 to indicate the starting point of text definition.

Menu Press F8 to exit the Text Program and return to the Main Menu.

To cancel any operation (printing, selecting, saving, loading, etc.), press BREAK (SHIFT PAUSE).

Using the Command Keys in TEXT

The Command Keys have the same definition in all of the Application Programs. See Table 6-1.

Command Key Definitions	
Key	**Operation**
PASTE	Allows you to insert ("Paste" in) text that has previously been COPYed or CUT.
LABEL	Displays the current definitions of the Function Keys.
PRINT	Prints on a printer whatever is currently on the Display.
BREAK	To cancel any operation, press BREAK (SHIFT PAUSE).

Table 6-1

To create a Text file:

1. When the Main Menu is displayed, position the Cursor over the word `TEXT` and press ENTER.
2. The Menu disappears and you are asked:

   ```
   File to edit?
   ```

   Type in a file name (no more than 6 characters) and press ENTER.

   (When you eventually return to the Main Menu, you'll see that the Text Program will have added the ''extension'' .DO to the file name you typed in.)
3. The Display will clear and you can begin typing.

To close a Text file:

Once you've created a Text file, ''close'' it (returning to the Main Menu) before turning the power off.

To close the file, press F8.

To open a previously created Text file:

When a Text file has been created and properly closed, the file name you assigned to the Text file (plus the extension .DO) will be displayed on the Main Menu.

To ''open'' that file, position the Cursor over the file name (use the arrow keys) and press ENTER.

To delete a Text file from the Main Menu:

When you no longer need a Text file, you can delete or ''kill'' it from the Main Menu. To do so:

1. Enter BASIC by positioning the Cursor over the word `BASIC` (use the arrow keys) and press **ENTER**.
2. When the BASIC OK prompt appears, type **KILL** *"filename"* and press **ENTER** where *filename* is the name of the Text file which *must* include the extension .DO. Don't forget to include the quotation marks around the file name.
3. The OK prompt will return when the Text file has been deleted. Return to the Main Menu by pressing **F8**.

To print a Text file on a printer:

If you have the Model 100 connected to a parallel printer (see **''Connecting the Model 100 to a Printer''** in Appendix A for details), you can either print out the entire file or print out only what appears on the Display.

To print out the entire Text file, press **SHIFT** **PRINT**. The Model 100 will ask you `Width?` and display the current width setting. If you wish to change this, type in a number between 10 and 132 which specifies the column width you want on the printed paper.

To print out only the part of the Text file which appears on the Display, press **PRINT**.

Text Editing

Once you've created or opened a Text file, you can edit (add to or delete from) that file:

- A character at a time

or

- A ''block'' of characters all at once

To insert a single character:

Simply position the Cursor (using the arrow keys) to the point where the new text is to be inserted. Begin typing the new text. The existing text will automatically shift to the right one space for every character you type.

To delete a single character:

Position the Cursor (using the arrow keys) directly on top of the character you wish to delete. Press **DEL** (**SHIFT** **BKSP**).

Defining Blocks of Text

Defining blocks of characters enables you to move, delete, or duplicate words, lines, or entire files. Block definition is the fastest way to perform a variety of text editing tasks.

How do you know if a block of text has been "defined?" When text has been defined, it is shaded (or displayed in "reverse video").

Remember! A block of text can be a character, a word, a line, everything above the Cursor, or everything below the Cursor. In this section, we'll first show you how to define a block, then we'll show you what to do with it.

To define a block of text:

1. Position the Cursor to the "start" position of the text to be defined.
2. Press F7 (Select).
3. Define the block:
   - To include just one character, press →.
   - To include the entire word, press SHIFT →.
   - To include the entire line, press CTRL →.
   - To include all of the text to the end of the file, press CTRL ↓.
   - To include all of the text to be beginning of the file, press CTRL ↑.
   - To include a specific word somewhere later in the file, press F1. The prompt `String:` will appear at the bottom of the Display. Type in the word at which block definition is to end and press ENTER.

If you've defined more text than you intended, simply "back up" the Cursor with ←.

To cancel a block definition operation, press BREAK (SHIFT PAUSE).

Manipulating Blocks of Text

Once a block of text has been defined (with F7 and the Cursor Movement Keys), it can be:

- Deleted, using the F6 (Cut) key.
- Moved, using the F6 (Cut) and PASTE keys.
- Duplicated, using the F5 (Copy) and PASTE keys.

Text blocks which have been defined can be moved or copied to other places in the current file or inserted into files other than the current file.

To delete a block of text:

Define it in one of the manners described above; then simply press the F6 key. All of the text that was defined ("shaded") will disappear from the Display.

To move a block of text:

Define it in one of the manners described above; then press the **CUT** (F6) key. All of the text that was defined (''shaded'') will disappear from the Display.

If you want to move the text block to another file, press **MENU** (F8) to close the existing file, then open or create a new file.

Next, position the Cursor to the point where you want the text moved (in the current file or a different one) and press the PASTE key.

The previously defined text will reappear on the Display starting at the Cursor position.

To duplicate a block of text:

Define a block of text in one of the manners described above; then press **COPY** (F5). The ''shade'' will then disappear from the Display but the defined text itself will remain.

If you want to copy the text block to another file, press **MENU** (F8) to close the existing file, then open or create a new file.

Next, position the Cursor to the point where you want the text to be copied (in the current file or a different one) and press PASTE.

The previously defined text will reappear on the Display starting at the Cursor position.

Using the Schedule Organizer Program (SCHEDL)

The Schedule Organizer program (SCHEDL) lets you locate names, addresses, phone numbers, and any other information you may need to know. This information is stored in a special Text file which you must assign the name of NOTE. (When you eventually return to the Main Menu, you'll see that the program will have added the ''extension''. DO to word NOTE.)

There can be only one NOTE.DO file listed on the Display at a time.

Using the Function Keys in SCHEDL

The Function Keys F1 - F8 have unique definitions when you're using the SCHEDL program.

These unique definitions appear at the bottom of the Display and will look like Figure 6-2.

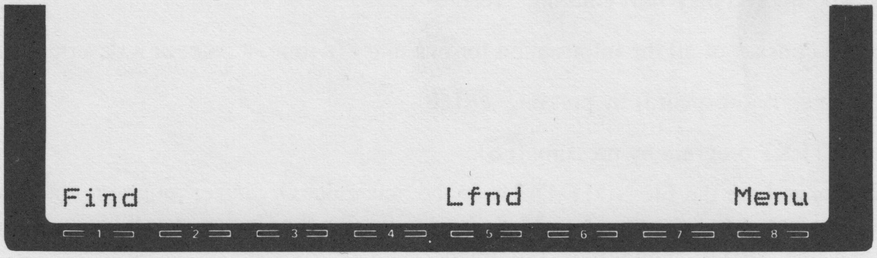


Figure 6-2. SCHEDL Function Key Definitions

Pressing LABEL causes the definitions to disappear from the Display.

Find Pressing F1 allows you to "find" a specified item in the NOTE file. Press F1 and type in the item (letters or numbers) you want to look for. The program will then Display the item (or items) which match the letters or numbers you typed.

F2 is not used by the SCHEDL Program.

F3 is not used by the SCHEDL Program.

F4 is not used by the SCHEDL Program.

Lfnd F5 works exactly like F1 except the information is printed on a printer (if one is connected) instead of the Display.

F6 is not used by the SCHEDL Program.

F7 is not used by the SCHEDL Program.

Menu Press F8 to exit the SCHEDL Program and return to the Main Menu.

To cancel any operation, press BREAK (SHIFT PAUSE).

Using the Command Keys in SCHEDL

The Command Keys have the same definition in all of the Application Programs. See Table 6-2.

Command Key Definitions	
Key	**Operation**
PASTE	Allows you to move or insert text that has previously been COPYed or CUT.
LABEL	Displays the current definitions of the Function Keys.
PRINT	Prints on a printer whatever is currently on the Display.
BREAK	To cancel any operation, press BREAK (SHIFT PAUSE).

Table 6-2

The steps necessary to create and use SCHEDL are:

1. Enter the TEXT program (at the Main Menu) by positioning the Cursor over the word `TEXT` and pressing ENTER. When you are asked for a file name, type `NOTE` and press ENTER. This creates a file called NOTE.DO.
2. At this point, you may start entering "records."

   A record consists of all the information for one entry (a time, a date, or a description).

   End each entry (or record) by pressing ENTER.
3. Exit the TEXT program by pressing F8.
4. To access and use the file NOTE.DO from the Scheduler Organizer program, enter the Scheduler Program from the Main Menu by positioning the Cursor over the word `SCHEDL` and pressing ENTER. You will be prompted:

```
Schd:
```

5. To locate a date, label, or other item, press **FIND** (F1). The Screen will display:

   `Schd: Find`

   Now type in the item you wish to find and press ENTER. If the item is found, the entire record is displayed. If it is not found, the `Schd:` prompt returns.

   SCHEDL finds and displays every record in which the item you're searching for occurs. If the number of records, or the record associated with the search item is too long to fit on the Display, SCHEDL prompts you with:

   `More Quit`

   Pressing **MORE** (F3) shows you the next record, pressing **QUIT** (F4) returns the `Schd:` prompt.

6. If you have a printer connected to the Model 100, you can get a printed copy of the record. Press F5 instead of F1. It functions identically to F1, except that it prompts you with:

   `Schd: Lfnd`

To exit the Scheduler Program, press **MENU** (F8).

A Few Notes on SCHEDL . . .

- Try to keep the record format consistent. For example, you might list the date first, followed by the time, then the location, and finally a comment about the event.
- Separate the items of the record (the time from the date) either by a punctuation mark such as a comma or tab key, or by making each item a standard length.
- You may want to ''label'' the different records according to categories. For example, you might put a dollar sign before every record containing a bill due date, a phone symbol (press GRPH P) in front of important phone calls, and so on. This makes it easy to use the ''find'' feature. For example:

  $ 04/05/82 — 08:00 — Car Payment
  04/05/82 — 09:30 — Phone Home
  * 04/05/82 — 10:00 — Meet with the Chancellor
  04/05/82 — 16:00 — Bobby Sue's present
  04/05/82 — 21:00 — Phone Bob and Sue in Hawaii

- Pressing **FIND** (F1) and then ENTER (giving no search item), lets you ''thumb'' through the file a screenful at a time. This feature is also available for F5 — pressing F5 ENTER prints out the file a screenful at a time.
- Upper- and lowercase are ignored. You may enter JANUARY, january, or JanUarY, with equivalent results.
- You needn't enter the entire time or date for the search item — just enough to make it unique. You may find this feature useful if, for example, you want to see all important billing dates for the first nine days of the month — you might type F1 `10/0` ENTER, which returns all records with 10/0. You can also search for your labels, such as $ (for bills due).
- Updating the Schedule file is simple since you use the TEXT program. From the Menu, move the Cursor to `NOTE.DO`, and press ENTER. You now have full use of the TEXT features.

Using The Address Organizer Program (ADDRSS)

The Address Organizer Program lets you locate names, addresses, phone numbers, and other information. This information is stored in a special Text file which you must assign the name of ADRS. (When you eventually return to the Main Menu, you'll see that the program will have added the "extension" .DO to word ADRS.)

There can be only one ADRS.DO file listed on the Display at a time.

ADDRSS can be used strictly as an address organizer or it can be used by the TELCOM program to automatically dial a phone number. The only difference between the two applications is that address entries used by TELCOM must be in the order of:

- name
- phone number (with a colon before the number)
- address (which must be preceded by a colon)

Address items for ADDRSS can be in any order.

Using the Function Keys in ADDRSS

The Function Keys F1 - F8 have unique definitions when you're using the ADDRSS program.

These unique definitions appear on the bottom of the Display and will look like Figure 6-3.

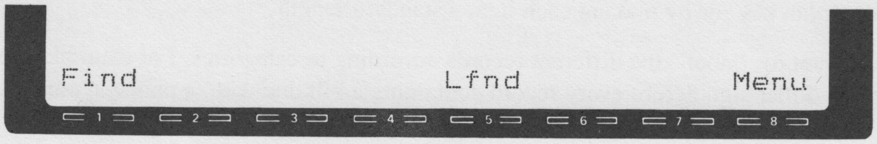


Figure 6-3. ADDRSS Function Key Definitions

Pressing LABEL causes the definitions to disappear from the Display.

Find Pressing F1 allows you to "find" a specified item in the ADRS file. Press F1 and type in the item (letters or numbers) you want to look for. The program will then display all the item (or items) which match the letters or numbers you typed.

F2 is not used by the ADDRSS Program.

F3 is not used by the ADDRSS Program.

F4 is not used by the ADDRSS Program.

Lfnd F5 works exactly like F1 except the information is printed on a printer (if one is connected) instead of the Display.

F6 is not used by the ADDRSS Program.

F7 is not used by the ADDRSS Program.

Menu Press F8 to exit the ADDRSS Program and return to the Main Menu.

To cancel any operation, press BREAK (SHIFT PAUSE).

Using the Command Keys in ADDRSS

The Command Keys have the same definition in all of the Application Programs. See Table 6-3.

Command Key Definitions	
Key	**Operation**
(PASTE)	Allows you to insert ("PASTE" in) text that has previously been COPYed or CUT.
(LABEL)	Displays the current definitions of the Function Keys.
(PRINT)	Prints on a printer whatever is currently on the Display.
(BREAK)	To cancel any operation, press (BREAK) ((SHIFT)(BREAK)).

Table 6-3

To use the ADDRSS Program:

1. Create the ADRS file by moving the Cursor (on the Main Menu) to the word `TEXT` and press (ENTER).

   When you're prompted for the name of the file to edit, type: `ADRS` and press (ENTER).

2. At this point, you may start entering "records."

   A record consists of all the data for one entry, for example, a name, an address, and a phone number may make up one record.

   End each entry (record) by pressing (ENTER).

3. When you have entered as many records as you want, return to the Main Menu by pressing (F8).

4. To access and use the file ADRS.DO from the Addresser Organizer program, enter the Addresser Program from the Main Menu by positioning the Menu cursor over the word `ADDRSS` and pressing (ENTER). You will be prompted:

   ```
   Adrs:
   ```

5. To locate an item (a name or address), press **FIND** ((F1)). The following message will appear:

   ```
   Adrs: Find
   ```

   Now type in the item you wish to find and press (ENTER). If the item is found, the entire record is displayed. If it is not found, the `Adrs:` prompt returns.

   ADDRSS finds and displays every record in which the item you're searching for occurs. If the number of records, or the record associated with the search item is too long to fit on the Display, ADDRSS prompts you with:

   ```
   More  Quit
   ```

   Pressing **MORE** ((F3)) shows you the next record, pressing **QUIT** ((F4)) returns the `Adrs:` prompt.

6. If you have a printer connected to the Model 100, you can get a printed copy of the record. Press (F5) instead of (F1). It functions identically to (F1), except that it prompts you with:

   ```
   Adrs: Lfnd
   ```

To exit the Addresser Program, press (F8).

A Few Notes on ADDRSS . . .

- Pressing **FIND** (F1) and then ENTER (giving no search item), lets you "thumb" through the file a screenful at a time. This feature is also available for **LFND** (F5).
- Upper and lowercase are ignored. You may enter JOHN, john, John, or jOhN, with the same results.
- You needn't enter the entire name or address for the search item — just enough to make it unique. You may find this feature particularly helpful if you can't remember a full name or address. For example, typing F1 `Bill` ENTER finds every record of a "Bill" in the file. Likewise, typing F1 `Elm` ENTER finds every record of anyone who may live on Elm Street (or for that matter, anyone named Elmer, Thelma, and so on).
- Try to keep the record format consistent. For example, you might list the name first, followed by the phone number, next by the address, and finally by a comment.
- Separate the items of the record (for example, the name from the address) either by some punctuation mark such as a comma or tab key, or by making each item a standard length. This will make manipulation of ADRS.DO easier.
- If you intend to use the Telecommunications Program for auto-dialing, pay particular attention to the way you enter phone numbers. For example:

```
Smith, John W.,:2305554993: 15434 Westwind Rd,
Pontiac, Michigan, 50994, BSIE, 1982 ENTER

Register, James C., :9195552421: 503 E. West St.,
Chapel Hill, North Carolina, 10045, BS Bio ENTER
```

  Colons should be placed before the telephone number to tell the Model 100 where to begin dialing. A second colon anywhere in the record (preferably before the address) will tell the Model 100 to stop dialing.

- Use optional "dashes" to separate the area code and prefix of a phone number.

Using the Telecommunications Program (TELCOM)

TELCOM works in two modes. First, in Entry Mode, it lets you automatically dial any number in the ADRS.DO file. Secondly, in Terminal Mode it permits computer-to-computer communications.

Once you've used Terminal Mode to access a host system, you may store incoming information for later viewing or printing (download), produce printouts of everything that appears on the Screen (echo), or transmit files previously prepared from TEXT (upload).

You can perform all these operations with the Function Keys F1 through F8.

Using the Function Keys in TELCOM

The Function Keys F1 - F8 have unique definitions in each mode (Entry and Terminal Modes).

On entering the TELCOM Program, you are automatically in Entry Mode. Figure 6-4 describes the Function Key definitions in this mode.

To enter Terminal Mode manually, press F4. Figure 6-5 describes the Terminal Mode Function Key definitions. You must have dialed the host's telephone number before pressing this key.

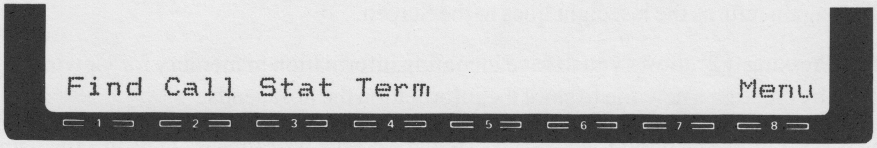


Figure 6-4. TELCOM Entry Mode Function Key Definitions

Pressing LABEL causes the definitions to disappear from the Display.

Find Pressing F1 allows you to "find" a specified item stored in the ADRS.DO file. Press F1, type in the item (letters or numbers) you want to look for, and press ENTER. The program will display the name and telephone number (up to the second colon) of the first record that matches the item you typed. You will be prompted for the next action:

```
Call More Quit
```

- **Call** Press F2 to dial the number displayed automatically.
- **More** Press F3 to find the next occurrence of a person's name.
- **Quit** Press F4 to "quit" the current name and telephone number. The prompt `Telcom:` will return and you may start over.

Call Press F2, type a phone number, and press ENTER to dial the number automatically.

Stat Pressing F3 lets you change the status ("setting") of the communication parameters (baud rate, word length, etc.) if necessary.

Term Pressing F4 lets you manually enter the Terminal Mode after calling a host computer's telephone number.

F5 is not used in TELCOM's Entry Mode.

F6 is not used by TELCOM's Entry Mode.

F7 is not used by TELCOM's Entry Mode.

Menu (F8) Press F8 to exit the TELCOM Program and return to the Main Menu.

To cancel any operation, press BREAK (SHIFT PAUSE).

Once in the Terminal Mode, the Function Keys are redefined and appear on the bottom of the Display.

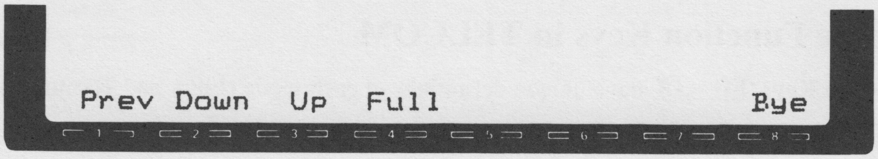


Figure 6-5. Terminal Mode Function Key Definitions.

Prev In Terminal Mode, the Display shows eight lines at a time. You may think of these eight lines as the bottom half of a page of text. To view the previous eight lines, it is only necessary to press F1. Pressing F1 again returns the last eight lines to the Screen.

Download Pressing F2 allows you to save incoming information in memory for viewing or printing later by creating a new file to store the information the host sends.

When you press F2, the prompt: `File to Download?` will appear. Type in a file name and press ENTER. To stop downloading, press F2 again.

Upload F3 allows you to send information that has been previously prepared in the TEXT Application Program to a host system.

When you press F3, the prompt: `File to Upload?` will appear. Type the name of the file which you assigned to the file you wish to send to the host and press ENTER. When the prompt: `Width:` appears, type in a number between 10 and 132 and press ENTER.

Full Pressing F4 lets you switch between Full and Half Duplex.

Most host systems require you to use Full Duplex. This means that any character you type is sent to the host before it appears on your Model 100 Display. If the characters you type are the same ones that appear on the Display, good communication with the host has been established.

Half Duplex, on the other hand, shows what you type directly on your Model 100 Display. This means that you have no way of knowing if the host received the same characters. (''Noisy'' telephone lines are sometimes the cause for this.)

Echo F5 enables you to obtain a printout (or ''hard copy'') of incoming information (if a printer is connected).

Wait This appears over F6 when XON/XOFF is enabled and a XOFF is sent to the Model 100. This halts the output from the Model 100. You can cancel ''wait'' by pressing BREAK but the host may lose data.

F7 is not used in Terminal Mode.

Bye Pressing F8 exits Terminal Mode and disconnects (or ''hangs up'') the telephone lines.

When you press F8, the prompt: `Disconnect?` will appear. At this time you may decide whether to terminate communications by pressing Y (for yes) or N (for no) and then ENTER.

Important Note! Pressing F8 followed by Y is necessary for the Model 100 to ''release'' the telephone line.

Using the Command Keys in TELCOM

The Command Keys have the same definition in all of the Application Programs and in both TELCOM operational modes. See Table 6-4.

Command Key Definitions	
Key	**Operation**
(PASTE)	Allows you to insert ("PASTE" in) text that has previously been COPYed or CUT.
(LABEL)	Displays the current definitions of the Function Keys.
(PRINT)	Prints on a printer whatever is currently on the Display.
(BREAK)	To cancel any operation, press (BREAK) ((SHIFT)(PAUSE)).

Table 6-4

Before using TELCOM, the Model 100 must be connected to the phone lines with a Modem Cable or an Acoustic Coupler (see Chapter 11 for details). The auto-dialing and auto log-on functions work only when the Computer is connected to modular phone lines using the built-in modem and the ***Model 100 Modem Cable***.

Entry Mode

When auto-dialing, TELCOM uses the names and phone numbers that you previously stored in the ADRS.DO file. This is a file created and updated from TEXT, where you may store:

- names
- phone numbers (with a colon before)
- address (which should be preceded by a colon)

For example:

```
Joe Dunn :555-1234: 317 Red River
```

To auto-dial a phone number in the ADRS.DO file:

1. Access TELCOM by moving the Cursor to `TELCOM` and pressing (ENTER).
2. Press **FIND** ((F1)). When prompted, type in the name of the person you want to phone, and press (ENTER).

After the name and number appear on the Screen, the ''new'' Function Keys options are displayed:

```
Call More Quit
```

3. Press **CALL** ((F2)).
4. Pick up the receiver before auto-dialing is complete.

   The prompt `Calling` will appear followed by the person's name and the telephone number digits will appear as they are automatically dialed.

   If a person's name appears more than once in the ADRS.DO file, you may find its next occurrence by pressing **MORE** ((F3)).

   If you want to find a different name and telephone number, press **QUIT** ((F4)). The prompt `Telcom:` will return and you may start over.

   If you typed in the wrong name (when prompted by (F1)), press (F4) to end the selection process, then start over.

To cancel a call while dialing is in progress, press (BREAK) ((SHIFT)(PAUSE)).

Terminal Mode

Before communicating with a host computer system, you must enter Terminal Mode. This may be done in two ways — automatically and manually.

Entering Terminal Mode Automatically

This method requires you store the host's name and phone number in the ADRS.DO file and that you add the symbols < > to it. For example:

CompuServe :5551234< >:

Once you've stored the phone number in the ADRS.DO file:

1. Position the Cursor over the word `TELCOM` and press (ENTER).
2. Set the **ANS/ORIG** Switch to **ORIG**.
3. Press **FIND** ((F1)), type the host's name, and press (ENTER).
4. Press **CALL** ((F2)). (You don't have to pick up the receiver.)

The Computer will produce a high-pitched tone as it enters the Terminal Mode and the definition of the Function Keys will be displayed to show the new options.

Entering Terminal Mode Manually . . .

With this method, you don't have to store the host's number in the ADRS.DO file. Simply:

1. Position the Cursor over the word `TELCOM` and press (ENTER).
2. Set the **ANS/ORIG** Switch to **ORIG**.
3. Lift up the telephone receiver and dial the host system's phone number.
4. When you hear a high-pitched tone, press **TERM** ((F4)).
5. Hang up the telephone receiver.

The Computer will produce a tone and the Display will list the new Function Key options. (See Figure 6-5 earlier in this section.)

Once you've entered Terminal Mode (automatically or manually), refer to your information services user's guide to log-on (i.e.,entering password and user ID) and communicate with the host.

Logging-on to an Information Service Automatically

Every time you establish communications with an information service, you must complete the necessary log-on information. This usually involves answering the User ID and Password prompts correctly. This may vary with some services.

TELCOM lets you log-on to a system automatically. To do so, you must first create a **Log-on Sequence** in TEXT. This consists of anticipating the host's log-on prompts, then sending your responses. This sequence must then be stored inside the symbols < > in the ADRS.DO file. This is called an Automatic **Log-on Sequence**.

Table 6-5 describes the commands needed to create a Log-on Sequence.

Command	Function
=	Pause for two seconds.
?	Wait for a specified character.
^	Causes the next character to be sent as a "control" character (i.e., ^M is the same as ENTER).
!	Send a specified character.

Table 6-5

The following sample Log-on Sequence can serve to illustrate how a Log-on Sequence works. You should consult your information service user's guide for specific instructions.

```
<=^C?U1234,567^M>
```

1. Start by typing the less than symbol (<).
2. You should use a pause (=) only when the first action expected by host is to receive a character from you.

   = means "pause for two seconds" and is used to establish a good phone link with the host.

   Pausing for two seconds allows the first character to be received by the host more reliably.
3. ^C causes a "control-C" to be sent to the host. This is required by CompuServe for instance.
4. Next type a question mark (?) which means "wait for the character that follows the question mark."

   This tells the Model 100 to look for a unique character in the prompt that will be sent from the information service to the Model 100.
5. Then type the character the Model 100 is to look for. In this case it is a capital **U**. This is a unique character in the prompt **User ID:** which is sent by the host.

   Notice that only one character from the prompt is necessary.
6. Now type in the response you would normally send to the host. In this case, it is the User ID: **1234,567**.
7. At the end of the User ID, the host normally looks for you to press ENTER (a carriage return). For the Log-on Sequence, press SHIFT 6 M which displays **^M**. This causes the Model 100 to send a **^M** character which is the same as ENTER. The host can then acknowledge your response.
8. If the next prompt from the host is a password, repeat the process but use the new prompt and response.

   For instance, type a **?**, a **P** (a unique character in the prompt), the password itself (**PASSWORD** in this case), followed by another **^M**. Close the Log-On Sequence with a greater than symbol (>).

Follow the steps in this sample to create other Log-on Sequences. With some information services, you may have to include additional information in the Log-on Sequence. Consult your information service user's guide.

Be sure to store the Log-on Sequence in the ADRS.DO file together with the host's name and phone number. For example:

```
CompuServe :5551234 <=^C?U1234,567^M?PPASSWORD^M>:
```

Note: The command **!** is only used when your responses to a prompt includes the symbols **?** or **=**. **!** lets the Model 100 distinguish between commands such as **? =**, and responses which include the same symbols.

Communications Parameters

Communications parameters are a series of conventions for transmitting and receiving information when one computer is linked to another computer.

When communicating with another computer, the parameters of the Model 100 must match the parameters of the other computer before communication can take place (i.e., they must both speak the same "language").

TELCOM has predefined communication parameters. See Table 6-6 for those settings.

TELCOM Start-Up Communications Parameters	
Meaning	**Start-Up Setting**
Baud Rate	M (300 baud)
Word Length	7 (bits)
Parity	I (Ignore Parity)
Stop Bit	1
Status	E (Enable)
Dial Pulse Rate	10 pps

Table 6-6

If you need to change the Model 100's communication parameters:

When changing communication parameters, you must type in each and every selectable parameter even if you don't want it changed. To leave a parameter at its current status ("unchanged value"), simply type in the current value as displayed on the Screen.

1. Access TELCOM.
2. Press (F3). When you do this, the word `Stat` appears next to `Telcom` on the Display.
3. Type the new communication parameters in the following order:
    1. Baud Rate
    2. Word Length
    3. Parity
    4. Stop Bit
    5. Line Status
    6. Pulse Rate (pulse rate must be preceded by a comma).

For instance, if the current status of TELCOM program is:

M7I1E,10

and you want to change the parity to Even, type:

```
M7E1E,10 (ENTER)
```

You can use any of the allowable values listed in Table 6-7.

Modei 100 Communications Paramenters		
	You Type:	**For:**
Baud Rade	M	"modem" (300)
	1	75 baud
	2	110 baud
	3	300 baud
	4	600 baud
	5	1200 baud
	6	2400 baud
	7	4800 baud
	8	9600 baud
	9	19200 baud
Word Length	6	6 bits
	7	7 bits
	8	8 bits
Parity	I	Ignore parity
	O	Odd parity
	E	Even parity
	N	No parity
Stop Bit	1	1 stop bit
	2	2 stop bit
Line Status **Line**	E	Enable
	D	Diasble Disable
Pulse Rate	10	10 pps
	20	20 pps

Table 6-7

***Note:** The Model 100 uses 300 baud when the built-in modem is in use. If you use a number to set the baud rate, even if that number is 3 (for 300 baud), the modem becomes disabled. The RS-232C Interface then becomes enabled. For this reason, always select the letter **M** whenever the built-in modem is to be used.

**Line Status is the XON/XOFF status. To send a XON ''manually,'' type CTRL Q; to send XOFF, type CTRL S.

PART II / THE MENU OPTIONS

The Model 100 has five built-in Application Programs that perform a variety of functions. For instance, you can write your own programs in BASIC, create, alter, and manipulate text for memos or letters, or keep a record of your expense account items and appointments. In addition, you can store and retrieve phone numbers for automatic dialing or computer-to-computer communications over the telephone.

This section of the manual will describe in detail how to use the Applications Programs you were introduced to in Part I.

Chapter 7 will start with a brief, general discussion of the common elements of the programs, including file names (syntax and extensions) and other background information.

Then, in Chapter 8, we'll begin with Text, the Application Program you'll probably use more than any other. This will be followed by a description of each of the other programs — Scheduler Organizer (Chapter 9), Address Organizer (Chapter 10), and Telecommunications (Chapter 11).

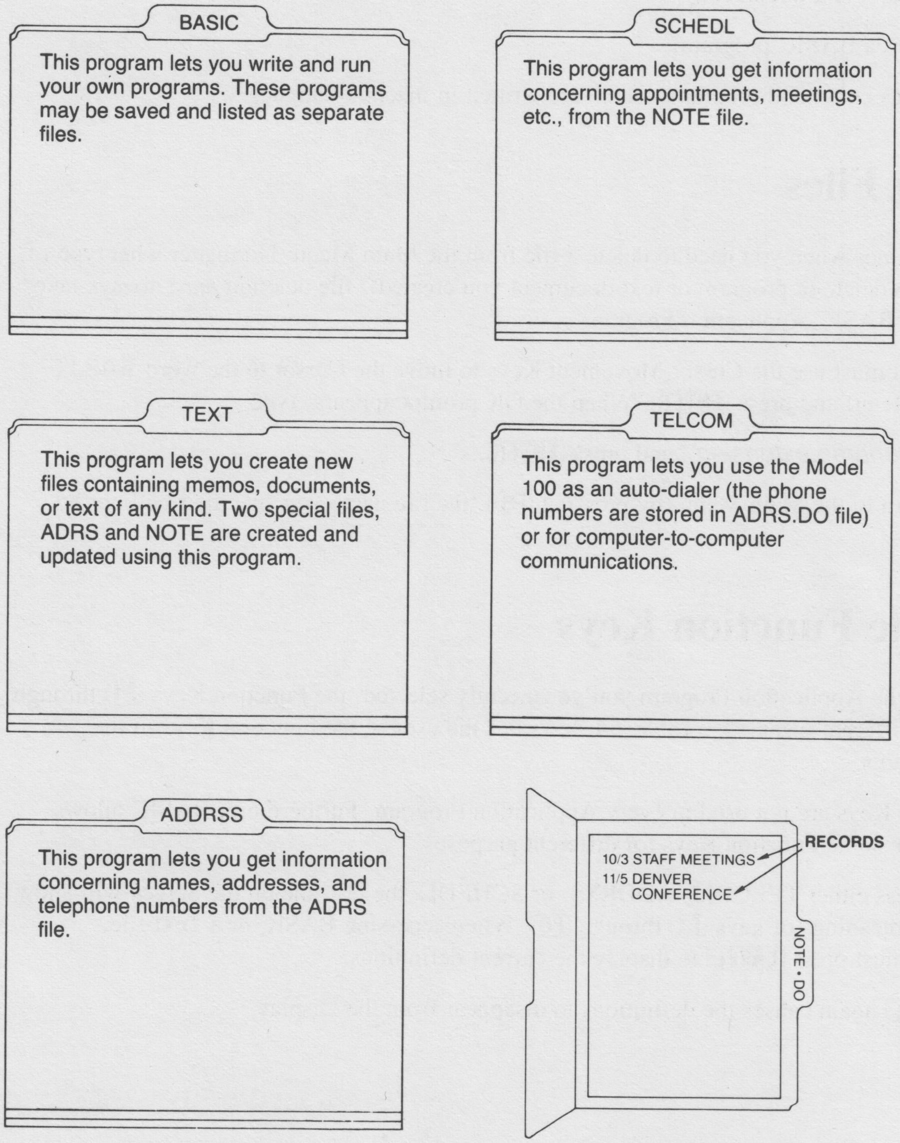
It's important to note that the Application Programs interact with each other. For instance, you can transmit (over the telephone) a document created in TEXT via TELCOM. It's also possible to sort out or rearrange the Scheduler (SCHDEL) or Addresser (ADDRSS) by writing a program in BASIC. Appendix C provides sample sessions that demonstrate a few ways in which the Application Programs interact.

7 / Menu Overview

When you power-up the Model 100, the Main Menu shows you the names of all ''files'' in the Computer. Think of these Computer files as ordinary file folders that may contain either programs or text.

Although the Model 100 is delivered with five built-in files containing the Application Programs, you can create new program and text files. When you do so, the names you assign these files will also appear on the Main Menu.

Each file may contain many ''records.'' A record is simply an item within the file. A name, an address, and a telephone number, for instance, constitute one kind of record.



BASIC

This program lets you write and run your own programs. These programs may be saved and listed as separate files.

SCHEDL

This program lets you get information concerning appointments, meetings, etc., from the NOTE file.

TEXT

This program lets you create new files containing memos, documents, or text of any kind. Two special files, ADRS and NOTE are created and updated using this program.

TELCOM

This program lets you use the Model 100 as an auto-dialer (the phone numbers are stored in ADRS.DO file) or for computer-to-computer communications.

ADDRSS

This program lets you get information concerning names, addresses, and telephone numbers from the ADRS file.

Figure 7-1. Model 100 File and Records

File and Program Names

When you create a file or save a program, you must assign it a name. That name *cannot* exceed six characters in length. You can assign Text files or BASIC programs any name you wish.

The Address file (where you store names, telephone numbers, and addresses) *must* be assigned the file name **ADRS**.

In the same sense, the Scheduler file (where you store information about dates and times for meetings, appointments, etc.) *must* be assigned the file name **NOTE**.

When the file or program name you have specified appears on the Main Menu, it will be followed by a ''file extension'' which is automatically assigned by the Model 100.

The file extension differentiates documents from programs, and consists of the letters .DO, .BA, or CO.

- **.DO** identifies a text document.
- **.BA** identifies a BASIC program.
- **.CO** identifies command programs which are written in machine-language.

Deleting Files

There will be times when you need to delete a file from the Main Menu. No matter what type of file you wish to delete (a program or text document you created), file deletion *must* always take place using the BASIC Application Program.

This means you must use the Cursor Movement Keys to move the Cursor to the word `BASIC` (on the Main Menu) and press **ENTER**. When the OK prompt appears, type:

KILL *''filename.extension''* and press **ENTER**.

When you return to the Main Menu (by pressing **F8**), the file name you specified will not be listed.

Using the Function Keys

Depending on the Application Program you've currently selected, the Function Keys **F1** through **F8** can have different meanings. Table 7-1 describes the uses of the Function Keys in the different programs.

Some Function Keys are not used in every Application Program. Furthermore, BASIC allows you to re-define some Function Keys for different purposes.

When you access either TELCOM, ADDRSS, or SCHEDL, the last line on the Screen will show the ''current'' meanings of keys **F1** through **F8**. When accessing BASIC or a Text file, however, you must press **LABEL** to display the current definitions.

Pressing **LABEL** again causes the definitions to disappear from the Display.

Program	F1	F2	F3	F4	F5	F6	F7	F8
BASIC	Files	Load	Save	Run	List	—	—	Menu
TEXT	Find	Load	Save	—	Copy	Cut	Sel	Menu
TELCOM	Find	Call	Stat	Term	Echo	Wait	—	Menu
ADDRSS	Find	—	—	—	Lfnd	—	—	Menu
SCHEDL	Find	—	—	—	Lfnd	—	—	Menu

Table 7-1. Function Key Definitions

For your convenience, we'll refer to a Function Key by the operation it performs (SELECT, FIND, COPY, etc.) rather than by the identifying label of the key (F1, F2, etc.).

To use a Function Key, simply press it at anytime. In some cases you will be prompted to type in a file name (Load, Save, etc.) while in other cases you'll be asked to type in an item you want to search for (Find, etc.).

Note that **Lfnd** is the same as **Find** except the information you're looking for is printed out on a printer instead of on the Display. If you use Lfnd and a printer is not connected, the Model 100 will appear to do nothing. Press BREAK (SHIFT PAUSE) to return to normal operation. **Find** and **Lfnd** will only search "down" a file. That is, if the Cursor is positioned at the end of a file and you search for an item that is before it, the item will not be found.

Pressing F8 will always save the current file and return you to the Main Menu.

Using the Command and Cursor Movement Keys

The Command Keys (PASTE, LABEL, PRINT, and PAUSE) and the Cursor Movement Keys (←, →, ↑, and ↓) perform the same operations in all of the Application Programs.

8 / Text Preparation (TEXT)

The TEXT Application Program is a simple but powerful text preparation tool for word processing or any other application that requires you to create or manipulate text. This includes files prepared in TEXT that are used by the other Application Programs.

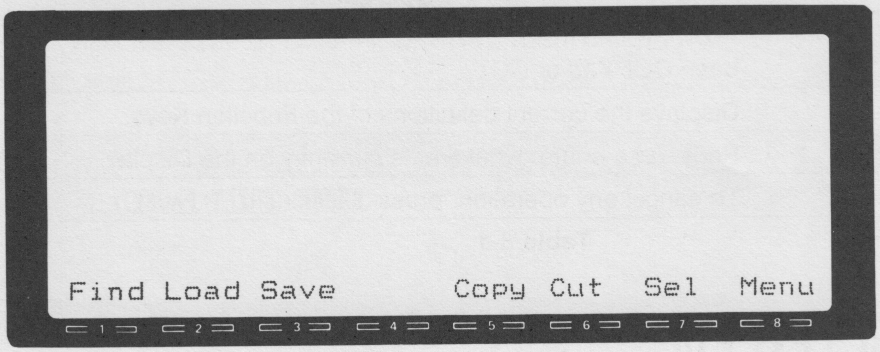
With TEXT, you can move, duplicate, or delete text that has been stored in a file. You perform these operations by using the **COPY** (F5), **CUT** (F6), and **SELECT** (F7) Function Keys in conjunction with the PASTE Command Key.

SELECT and the Cursor Movement Keys select or define blocks of text to manipulate (move, delete, or duplicate) at a later time. **COPY** and **CUT** takes whatever information you've "selected" and stores it in a special part of the Computer's memory for duplication or deletion.

Using the Function Keys in TEXT

The Function Keys F1 through F8 have more uses in Text than any other Application Program. In this section, we'll briefly describe the functions. The rest of the chapter will provide examples that show you how to use the Function Keys to make operating your Model 100 both easy and efficient.

To see the functions these keys can perform, press LABEL and the bottom line of the Display will look like this:



Find Pressing F1 allows you to "find" a specified item within a file. Press F1 and type in the "string" (a sequence of any characters — letters or numbers) you want to look for. The Cursor will move to the first occurrence of what you specify.

Load Pressing F2 allows you to get information from a device (such as a cassette recorder) into the Computer.

This is particularly useful if you write documents in which some paragraphs are standardized or have the same content.

One way you might use this function in Text is to avoid retyping entire sections. Simply type the standardized paragraph once, store it on tape, and use F2 to load it into your current file.

Save Pressing F3 lets you store information into a device (such as a cassette recorder).

It's a good idea to save files and programs on tape to prevent inadvertent loss of valuable information. You may also find it necessary to save files on tape to free additional memory space in the Model 100.

(F4) is not used by the TEXT Program.

Copy Once text has been ''defined'' with the (F7) key and the Cursor Movement Keys, you can duplicate (''copy'') it by pressing (F5). See the appropriate section of this chapter for more details.

Cut Once text has been ''defined'' with the **SELECT** ((F7)) and the Cursor Movement Keys, you can delete (''cut'') it by pressing (F6). See the appropriate section of this chapter for more details.

Sel Press (F7) to mark the beginning of text to be ''defined,'' and use the Cursor Movement Keys to include the desired text. See the appropriate section of this chapter for more details.

Menu Press (F8) to exit the Text Program and return to the Main Menu.

To cancel any operation (printing, selecting, saving, loading, etc.), press (BREAK) ((SHIFT)(PAUSE)).

Using the Command Keys in TEXT

The Command Keys have the same definition in all of the Application Programs. See Table 8-1.

Command Key Definitions	
Key	**Operation**
(PASTE)	Allows you to insert ("PASTE" in) text that has previously been COPYed or CUT.
(LABEL)	Displays the current definitions of the Function Keys.
(PRINT)	Prints on a printer whatever is currently on the Display.
(BREAK)	To cancel any operation, press (BREAK)((SHIFT)(PAUSE)).

Table 8-1

Printing a Text File

If you have the Model 100 connected to a parallel printer (see **''Connecting the Model 100 to a Printer''** in Appendix A for details), you can either print out the entire file or print out only what appears on the Display.

To print out the entire Text file, press (SHIFT)(PRINT). The Model 100 will ask you `Width?` and display the current width setting. If you wish to change this, type in a number between 10 and 132 which specifies the column width you want on the printed paper.

To print out only the part of the Text file which appears on the Display, press (PRINT).

Using the Cursor Movement Keys in TEXT

The Cursor Movement (''arrow'') keys are used the same in all of the Application Programs. You can move the Cursor a character at a time by pressing the appropriate key or you can use a CTRL or SHIFT key combination to move the Cursor quickly. See Table 8-2.

	By itself:	SHIFT *key*	CTRL *key*
		Moves the Cursor to the:	
→	one character to the right.	beginning of the first word to the right.	right end of the current line.
←	one character to the left.	beginning of the first word to the left.	left end of the current line.
↑	one line up in the current column.	top of the Display in the current column.	beginning of file.
↓	one line down in the current column.	bottom of the Display in the current column.	end of the file.

Table 8-2

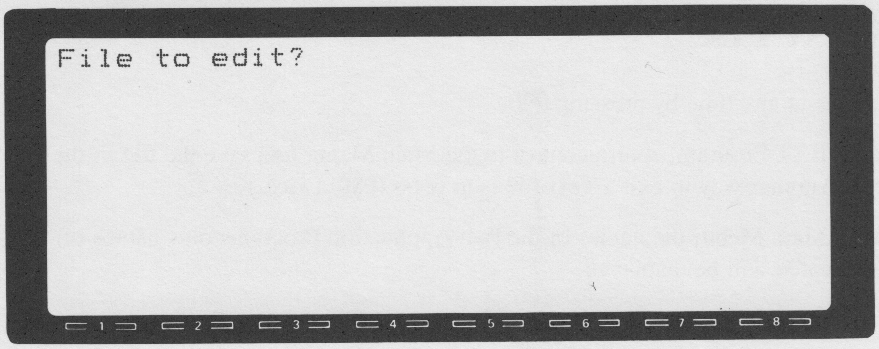
Creating a Text File

Before you can begin manipulating any text, you must create a file to store the information. That file is assigned a name when you create it. This file name appears on the Main Menu when the Model 100 is powered up and serves to identify and differentiate one document from another.

Remember that all Text files are automatically assigned the file name extension **.DO**.

To create a Text File:

1. Enter the TEXT Application Program by positioning the Cursor over `TEXT` and pressing ENTER or by typing `TEXT` ENTER.
2. The following message will then appear:



```
File to edit?
```

3. Type in the file name you wish to assign to the file and press ENTER. (Do not include the extension — *the program does this for you*.)

   Remember! Model 100 file names consist of no more than six characters (letters, numbers, or spaces). The program automatically assigns the extension.

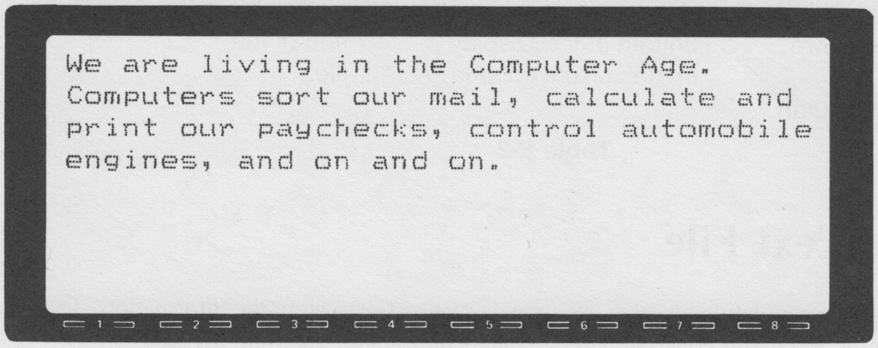
   For instance:

   ```
   File to edit? TEST1 ENTER
   ```

Immediately after you have entered the file name, the Display will clear (with the exception of the Cursor which remains on the upper-left corner). The Text file has been created and assigned a file name; you may begin entering information.

As you type, you do not have to press ENTER at the end of each line. If a word has more characters than spaces remaining on the line, that word will be displayed on the next line. TEXT will not ''break'' a word into two parts.

For example, type in the following exactly as it appears below. Remember that when you reach the end of a line, you will not have to press ENTER to return to the beginning of the next line. The Cursor will automatically move to the next line.



```
We are living in the Computer Age.
Computers sort our mail, calculate and
print our paychecks, control automobile
engines, and on and on.
```

After typing the last word in the paragraph, press ENTER.

When the eight lines of the Display are full, scrolling begins and the top line on the Display will be ''pushed'' out of sight.

To see a line after it has scrolled off the Display, press ↑ until the line you need to see appears. (Then press ↓ to get back to the line you were working on.)

For long Text files, press CTRL ↑ to get to the beginning of the file and CTRL ↓ to get to the end of the file.

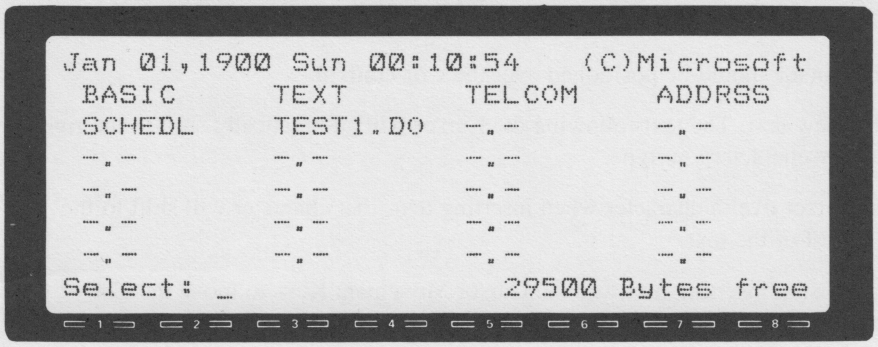
Closing a Text File

You can close a Text file at any time by pressing F8.

This will also exit the TEXT Program, return control to the Main Menu, and save the file in the Computer's memory. (Another way to exit a Text file is to press ESC twice.)

When you return to the Main Menu, the names of the five Application Programs plus names of all the files you have created will be displayed.

For instance, if you created the Text file TEST1 earlier, the Display will look like this after you press **F8**:



```
Jan 01,1900 Sun 00:10:54    (C)Microsoft
 BASIC      TEXT       TELCOM     ADDRSS
 SCHEDL     TEST1.DO   -.-        -.-
 -.-        -.-        -.-        -.-
 -.-        -.-        -.-        -.-
 -.-        -.-        -.-        -.-
 -.-        -.-        -.-        -.-
Select: _             29500 Bytes free
```

When you close and save a Text file, it is automatically assigned the file extension **.DO** for "document." (Note that .DO is also the extension for files ADRS and NOTE which are used by the ADDRSS, SCHEDL, and TELCOM Application Programs.) This file name extension indicates that the file is a document as opposed to a program.

To open (i.e., access) a Text file after it has been closed, move the Cursor over the file name by pressing **SPACEBAR** or the Cursor Movement Keys and press **ENTER**.

Or you can use the Command Line and type the file name. If you choose to use the Command Line, you must include the extension .DO.

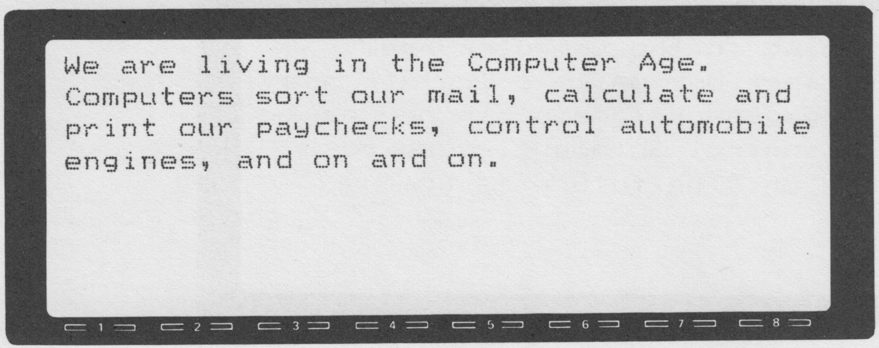
For instance, move the Cursor over `TEST1.DO`; then press **ENTER**. Or type: `TEST1.DO` **ENTER**. In both cases, the Text file you previously created will be opened.

Using the TEXT Editing Functions

The Text editing capabilities allow you to insert, delete, or relocate entire blocks of text either within a document or from one file to another.

Text Insertion and Addition

Inserting text is perhaps the simplest editing function. Assume you've created a file that contains the following text and assigned it the file name of TEST1:



```
We are living in the Computer Age.
Computers sort our mail, calculate and
print our paychecks, control automobile
engines, and on and on.
```

Now you want to insert the phrase "maintain our bank accounts" right after "calculate and print our paychecks."

To insert text into a Text file:

1. Position the Cursor (using the Cursor Movement Keys) at the point where you want to insert the new text.

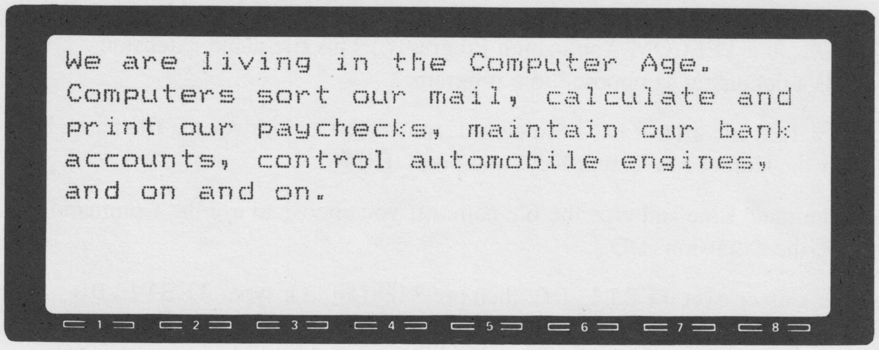
   In this case, the Cursor should be positioned over the **c** of **control**.

2. Begin typing the new text. The text following the Cursor will automatically shift to the right one space for every character you type.

If you position the Cursor over a character when inserting text, that character will shift to the right along with the rest of the text.

When you're ready to stop inserting text, use the Cursor Movement Keys to move the Cursor to another place in the text.

The paragraph will then look like this:



```
We are living in the Computer Age.
Computers sort our mail, calculate and
print our paychecks, maintain our bank
accounts, control automobile engines,
and on and on.
```

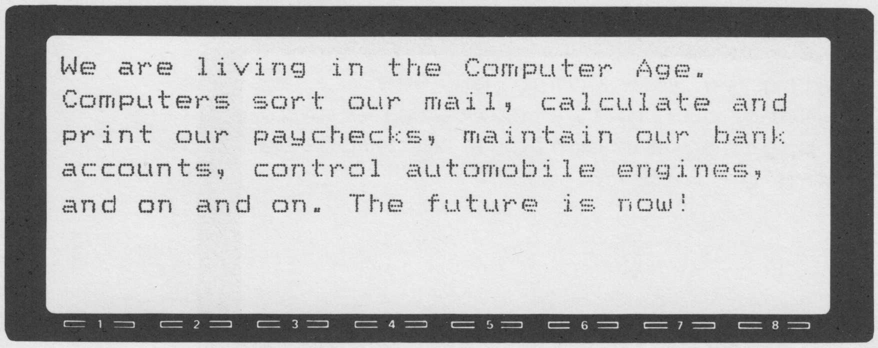
Text Addition

To add to a previously created Text file, simply position the Cursor to the end of the file and begin typing.

For instance, add the sentence "The future is now!" to our sample paragraph.

1. Press **CTRL** **↓** to move the Cursor to the end of the file.
2. Begin typing the additional information.

The paragraph will then look like this:



```
We are living in the Computer Age.
Computers sort our mail, calculate and
print our paychecks, maintain our bank
accounts, control automobile engines,
and on and on. The future is now!
```

Defining Blocks of Text

The TEXT Program allows you to ''define'' sections of text within a file, then move, duplicate, or delete that text.

To duplicate text, press the **COPY** (F5) Function Key; to delete text, press **CUT** (F6).

In both cases, the text you define and manipulate is stored in a special part of the Model 100's memory called the ''PASTE Buffer.''

Once text is stored in this area of memory, you can re-insert it elsewhere into the same or different file by simply positioning the Cursor and pressing the PASTE Command Key.

The PASTE Buffer is emptied of the ''current'' contents everytime you use either F5 or F6. In other words, if you copy a block of text named ''A'' and cut another block of text named ''B,'' the Buffer will contain ''B,'' not ''A.''

When defining a ''block'' of text, you can specify:

- An individual character.
- A line on the Display.
- A sentence.
- A paragraph.
- All text up to a particular word (specified by F1).
- All text below the Cursor.
- All text above the Cursor.

When you define a block of text, the defined block will appear in ''reverse video.'' This means that the background will be dark while the character will be light colored (just the opposite of the normal display).

General Steps Necessary to Define a Block of Text:

1. Move the Cursor to the ''start'' position of the text you wish to define.
2. Press **SELECT** (F7).
3. Begin specifying which characters, words, or lines you wish to include in the block of text by advancing the Cursor with the Cursor Movement Keys.

Press **CUT** (F6) or **COPY** (F5) to manipulate the defined block.

If you define more text than intended, simply backspace the Cursor by pressing ←.

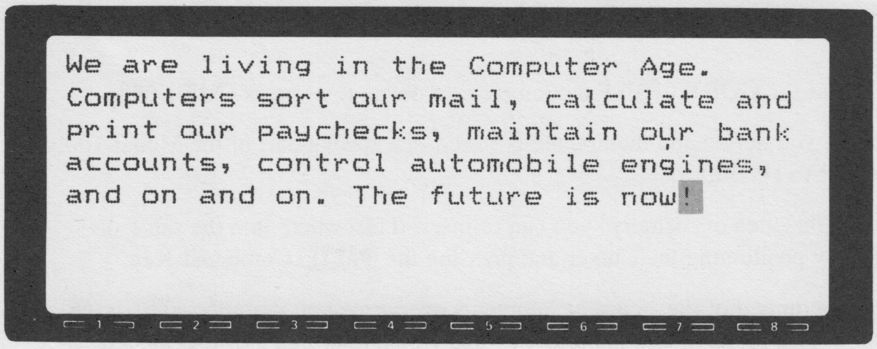
To cancel a block definition, press **BREAK** (**SHIFT** **PAUSE**).

Defining a Text Block

To include a character in a text block:

1. Position the Cursor on top of the character.
2. Press **SELECT** (F7).
3. Move the Cursor one space to the right by pressing →.
4. Delete or duplicate the character by pressing **CUT** (F6) or **COPY** (F5) respectively.

For instance, in the sample file you created earlier (TEST1), define the exclamation mark at the end of the paragraph.



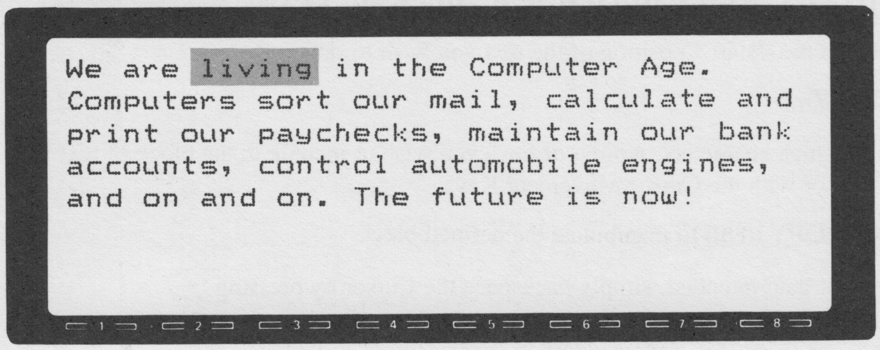
```
We are living in the Computer Age.
Computers sort our mail, calculate and
print our paychecks, maintain our bank
accounts, control automobile engines,
and on and on. The future is now!
```

Move the Cursor so that it is directly on top of !. One way is to press CTRL ↑ followed by ←. Press **SELECT** (F7) and move the Cursor one place to the right by pressing →. The exclamation mark will then appear in reverse video. Press **COPY** (F5), **CUT** (F6), or BREAK.

To include a word in a text block:

1. Position the Cursor on top of the first character of the word you wish to include.
2. Press **SELECT** (F7).
3. Press SHIFT →.
4. Delete or duplicate the word by pressing **CUT** (F6) or **COPY** (F5) respectively.

This time, assume the Cursor is positioned at the beginning of the file and you wish to define the word "living."



```
We are living in the Computer Age.
Computers sort our mail, calculate and
print our paychecks, maintain our bank
accounts, control automobile engines,
and on and on. The future is now!
```

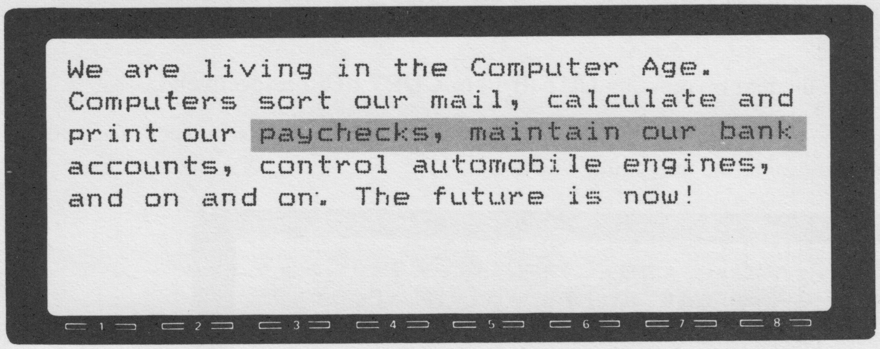
Press SHIFT → twice to move the Cursor to the beginning of the word "living." Press **SELECT** and then press SHIFT →. The word "living" will then appear in reverse video. Press **COPY**, **CUT**, or BREAK.

To include all characters from the Cursor to the end of a line:

1. Position the Cursor on top of the first character of the line you wish to include.
2. Press **SELECT** (F7).
3. Press CTRL →.

4. Delete or duplicate the text by pressing **CUT** (F6) or **COPY** (F5) respectively.

In this example, assume that the Cursor is at the beginning of the file and you wish to define all but the first two words (“print our”) of the third line of the paragraph.



```
We are living in the Computer Age.
Computers sort our mail, calculate and
print our paychecks, maintain our bank
accounts, control automobile engines,
and on and on. The future is now!
```

Move the Cursor to the first word of the third line by pressing ↓ twice. Then press SHIFT→ twice so that the Cursor is on top of the **p** in **paychecks**. Press **SELECT** (F7), followed by CTRL→ to define the rest of the line to the right. The defined text will be displayed in reverse video. You can then **CUT** or **COPY** it.

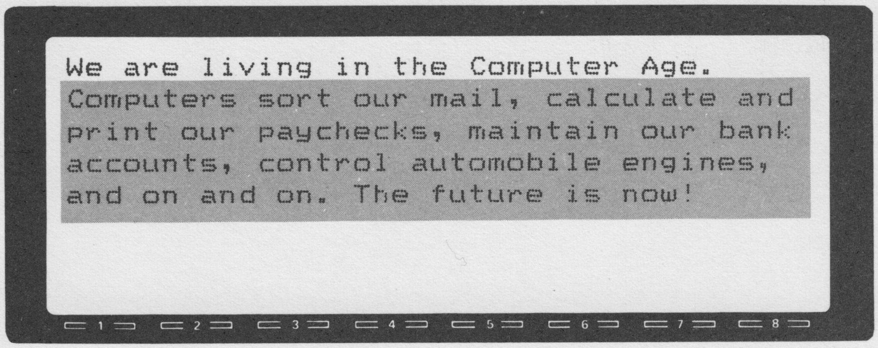
If you want to define all of the text from the Cursor to the left end (beginning) of the line, follow the same procedure but press CTRL← instead.

If you want to “undefine” the selected text, just press BREAK (SHIFT PAUSE).

To include the text from the Cursor to the end of the file:

1. Position the Cursor on top of the first character of the word where definition is to start.
2. Press **SELECT** (F7).
3. Press CTRL↓.
4. Delete or duplicate the text by pressing **CUT** (F6) or **COPY** (F5) respectively.

In this example, assume the Cursor is positioned at the beginning of the file, and you wish to define everything but the first sentence.



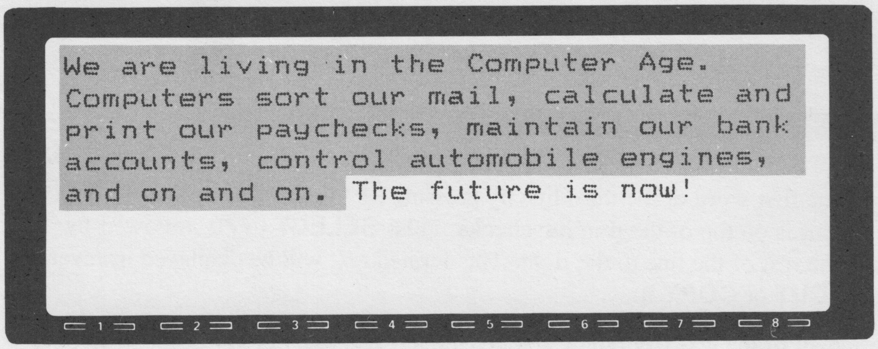
```
We are living in the Computer Age.
Computers sort our mail, calculate and
print our paychecks, maintain our bank
accounts, control automobile engines,
and on and on. The future is now!
```

Press ↓ to move the Cursor to the beginning of the second line. Press **SELECT** (F7) and CTRL↓. The defined text will be displayed in reverse video. You can then **CUT** or **COPY** it.

To include the text from the Cursor to the beginning of the file:

1. Position the Cursor on top of the first character of the word where definition is to start.
2. Press **SELECT** (F7).
3. Press CTRL ↑.
4. Delete or duplicate the text by pressing **CUT** (F6) or **COPY** (F5) respectively.

In this example assume the Cursor is positioned at the beginning of the last sentence "The future is now!," and you want to define everything above it.



```
We are living in the Computer Age.
Computers sort our mail, calculate and
print our paychecks, maintain our bank
accounts, control automobile engines,
and on and on. The future is now!
```

Press **SELECT** (F7) and CTRL ↑. The defined text will be displayed in reverse video. You can then **CUT** or **COPY** it.

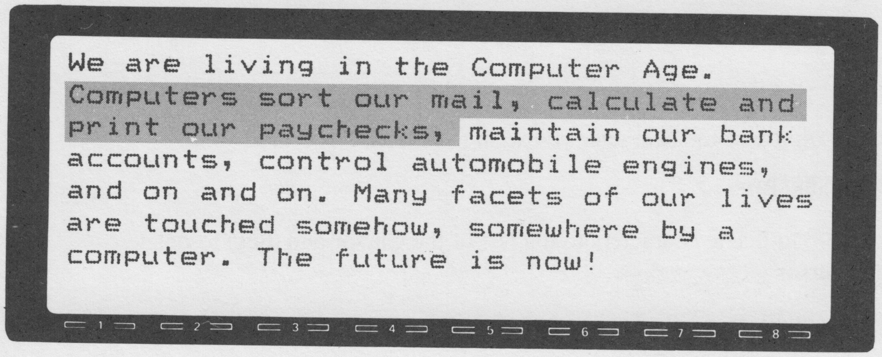
Using the FIND Command to Define a Block of Text

Another way to define text is to use the "find" command. This allows you to define text from the current Cursor position to a word or number which you specify by pressing **FIND** (F1) and typing the word. To use the find command to define a text block:

1. Position the Cursor on top of the first character you wish to include.
2. Press **SELECT** (F7).
3. Press **FIND** (F1).
4. When prompted by the word `String:`, type in the word or number where text definition is to end and press ENTER.

   All of the text between the Cursor and the first character you specified will be defined.
5. Delete or duplicate the text by pressing **CUT** (F6) or **COPY** (F5) respectively.

For instance, in the following paragraph, assume you wish to define the text from "Computers" to and including "paychecks."



```
We are living in the Computer Age.
Computers sort our mail, calculate and
print our paychecks, maintain our bank
accounts, control automobile engines,
and on and on. Many facets of our lives
are touched somehow, somewhere by a
computer. The future is now!
```

Position the Cursor on top of the **C** of **Computers** and press **SELECT** (F7). Next, press **FIND** (F1). When `String:` appears, type `maintain` ENTER since definition will end before the word **"maintain."**

The desired block will then be shaded and you can cut or copy it.

The TEXT Program also allows you to define text by pressing **SELECT** (F7) and moving the Cursor with the key-combinations of SHIFT followed by an arrow key or CTRL followed by an arrow key. See Tables 8-3.

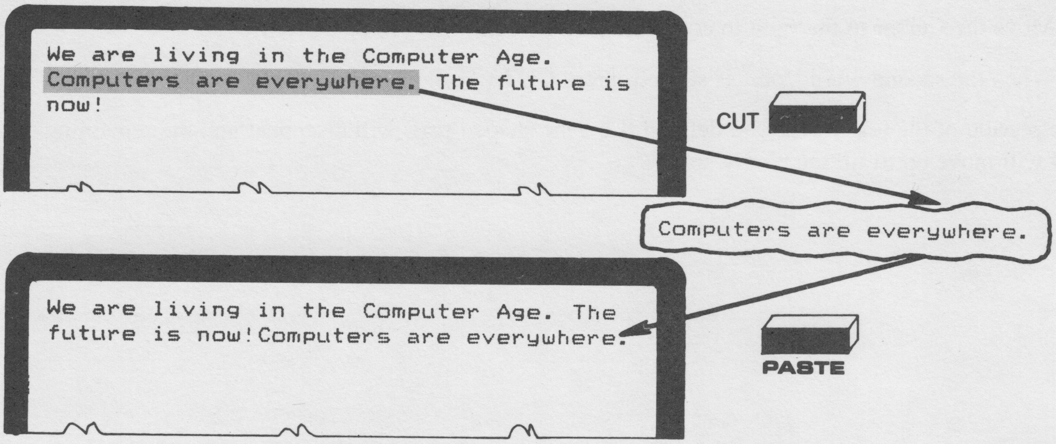
Deleting Text from a File (Cut)

There are two ways to delete text from a file.

One way is to delete individual characters from a file using the DEL (SHIFT BKSP) key. In this case, the character that the Cursor is on top of is permanently erased from the Text file.

Another way to delete text is to define it (character, word, line, above or below, etc.) and "cut" it by pressing the **CUT** Function Key.

Text deleted in this manner is stored in the PASTE Buffer and can be recalled if necessary until new text is stored in the Buffer. At that time, the text is permanently erased.



```
We are living in the Computer Age.
Computers are everywhere. The future is
now!
```

```
Computers are everywhere.
```

```
We are living in the Computer Age. The
future is now!Computers are everywhere.
```

Figure 8-1. Putting Text into the PASTE Buffer

To delete an undefined character:

1. Position the Cursor immediately to the right of the character you want to delete.
2. Press BKSP.

or

1. Position the Cursor on top of the character you wish to delete.
2. Press DEL (SHIFT BKSP).

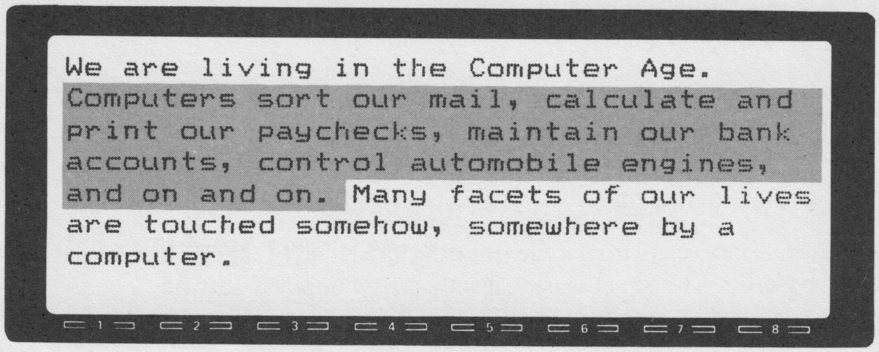
Continue pressing BKSP to delete characters to the left of the Cursor and DEL to delete characters under the Cursor.

To end Text Deletion, simply stop pressing the keys.

To delete a defined character or text block:

1. Define a block of text in one of the previously described manners.
2. Press **CUT** (F6).

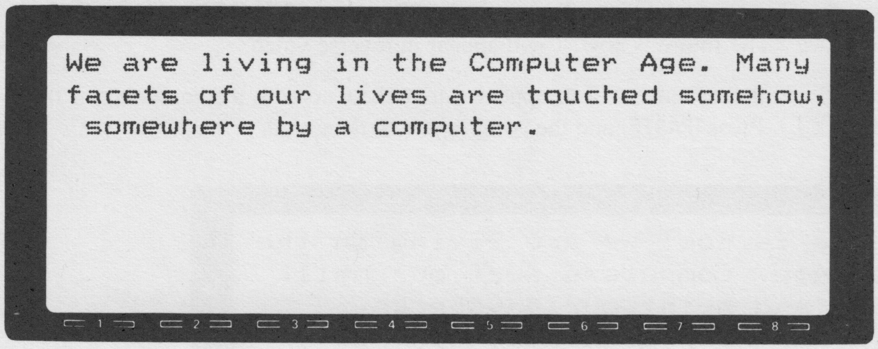
For example, in the following paragraph assume you want to delete the portion of the text beginning with "Computers sort our mail . . ." and ending with ". . . and on and on."



```
We are living in the Computer Age.
Computers sort our mail, calculate and
print our paychecks, maintain our bank
accounts, control automobile engines,
and on and on. Many facets of our lives
are touched somehow, somewhere by a
computer.
```

1. Position the Cursor on the **C** of the word **Computers**.
2. Press **SELECT** (F7).
3. Move the Cursor to the right to cover the entire block of text to be deleted.
4. When the second word "on" is shaded, press **CUT** (F6).

The portion of the text which you defined (i.e., the shaded part) will disappear and the remaining text will move up to fill the empty space.



```
We are living in the Computer Age. Many
facets of our lives are touched somehow,
 somewhere by a computer.
```

Remember! Text deleted in this manner is not permanently lost. Rather, it is stored in a PASTE Buffer where it remains until a new block of text is selected and deleted or duplicated.

Moving Text (Cut and Paste)

There may be occasions when a word, a phrase, or a block of text must be moved somewhere else within the document or even to a different file.

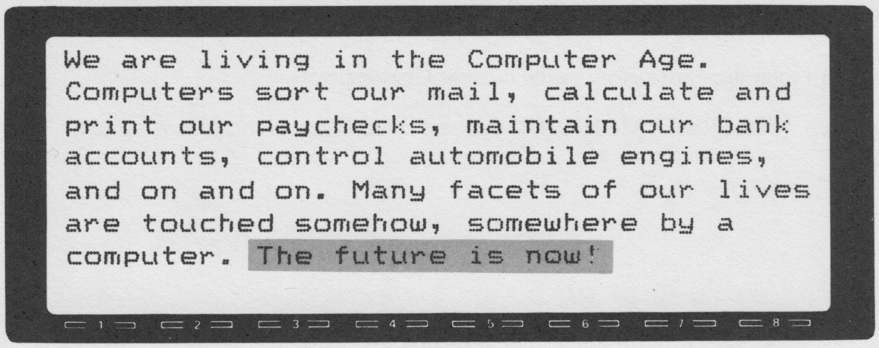
Pressing **CUT** (F6) will erase a defined block of text from the Display, store it in the PASTE Buffer, and allow you to insert the text elsewhere by positioning the Cursor and pressing the PASTE key.

The steps for doing this are exactly like those in Text Block Deletion with just a couple of extra steps added.

1. Position the Cursor at the beginning or the end of the text to be moved.
2. Press **SELECT** (F7).
3. Define the text block.
4. Press **CUT** (F6).
5. Move the Cursor to the place where you want to move the text (in the current file or another file).
6. Press PASTE.

   The defined text will then appear starting at the current Cursor position.

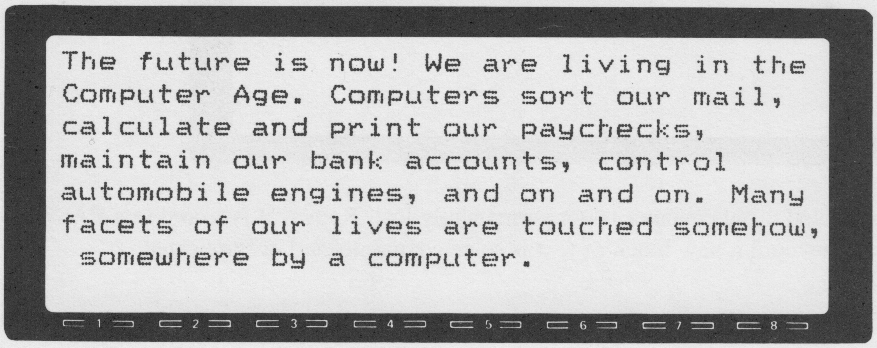
For example, move the last sentence in the following paragraph so that it is the first sentence.



```
We are living in the Computer Age.
Computers sort our mail, calculate and
print our paychecks, maintain our bank
accounts, control automobile engines,
and on and on. Many facets of our lives
are touched somehow, somewhere by a
computer. The future is now!
```

To do this, position the Cursor on top of the **T** in the word **The**. Press **SELECT** (F7), then CTRL →. The sentence "The future is now!" will appear in reverse video.

Now press **CUT** (F6) and the sentence will disappear. Move the Cursor to the beginning of the file by pressing CTRL ↑. Press PASTE and the paragraph will then read:



```
The future is now! We are living in the
Computer Age. Computers sort our mail,
calculate and print our paychecks,
maintain our bank accounts, control
automobile engines, and on and on. Many
facets of our lives are touched somehow,
 somewhere by a computer.
```

To relocate text to a different file, define and cut the text, exit the current file, and access the new file. Then position the Cursor on the proper position and press PASTE.

Duplicating Text (Copy and Paste)

Often times a phrase or statement appears repeatedly in a document. To avoid retyping the phrase or statement, use the **COPY** function (F5) in conjunction with PASTE.

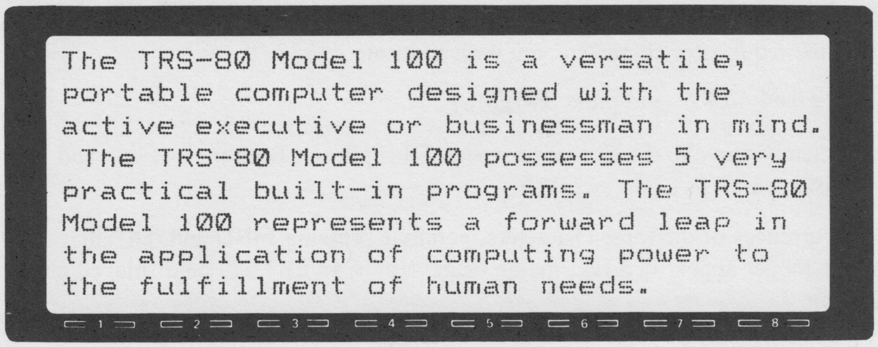
COPY is different from cut in that the defined text that you copy is not erased from the Display although it is duplicated in the PASTE buffer.

To duplicate text:

1. Position the Cursor at the beginning or the end of the text to be duplicated.
2. Press **SELECT** (F7).
3. Define the text block.
4. Press **COPY** (F5).
5. Move the Cursor to the place where you want to duplicate the text (in the current file or another file).
6. Press PASTE.

   The defined text will then appear starting at the current Cursor position.

For instance, the phrase "The TRS-80 Model 100" appears three times in the following text:



```
The TRS-80 Model 100 is a versatile,
portable computer designed with the
active executive or businessman in mind.
 The TRS-80 Model 100 possesses 5 very
practical built-in programs. The TRS-80
Model 100 represents a forward leap in
the application of computing power to
the fulfillment of human needs.
```

However, you only have to type it once. The other two times it is only necessary to use the Paste function.

To do this:

1. Position the Cursor at the beginning (or at the end) of the phrase to be repeated.
2. Define the text block. Note that the defined text will appear in reverse video.
3. Press **COPY** (F6).

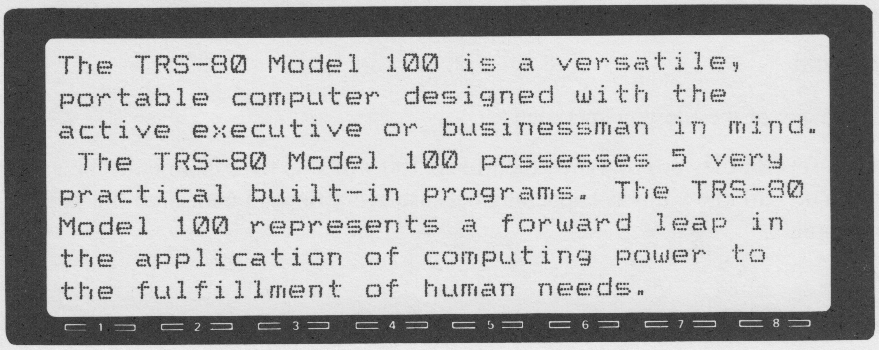
   The reverse video will immediately disappear — but not the text.
4. Continue typing until the phrase **The TRS-80 Model 100** needs to be repeated. Instead of re-typing it, simply press PASTE.

   The phrase **The TRS-80 Model 100** will then appear on the display.

Using the FIND Function

The **FIND** (F1) Function Key allows you to ''find'' or search for any character or group of characters in a Text file. Simply press **FIND** and, when prompted `String:`, type in the character you wish to search for. The Cursor will move to the first occurrence of the specified character(s). **FIND** will only search ''down'' a file, not from the bottom.

For instance, if you want to find the character string ''TRS-80 Model 100'' everytime it appears in the following paragraph:



```
The TRS-80 Model 100 is a versatile,
portable computer designed with the
active executive or businessman in mind.
 The TRS-80 Model 100 possesses 5 very
practical built-in programs. The TRS-80
Model 100 represents a forward leap in
the application of computing power to
the fulfillment of human needs.
```

Do this:

1. Using the Cursor Movement Keys, position the Cursor at the beginning of the file. (Placing the Cursor anywhere else in the file will start the search at that point.)

2. Press **FIND** (F1). When you do this, the message `String:` will appear on the last line of the Screen. This is the request for characters you wish to locate.
3. Type `TRS-80 Model 100` and press ENTER.

The Cursor will immediately move to the first occurrence of this phrase. Both the last line and the prompt will then disappear.

To find subsequent occurrences of the same characters, continue pressing **FIND** and ENTER. When the characters no longer appear in a file, the message `No match` will be displayed at the bottom of the Screen.

If you wish to search for different characters, simply press **FIND** and type the new characters when prompted. The old characters will disappear as soon as you press a key.

Using the LOAD Function

The F2 Function Key lets you load information from cassette tape into a Text file.

To load a program or data into a Text file:

Be sure the Model 100 is properly connected to a suitable cassette recorder (see Appendix A).

1. Access an existing file or create a new one.

   This depends on your purpose. For instance, you may want to insert a standard paragraph into an existing file. In this case, you would open the file in which you want to insert the information. On the other hand, you may want to load some information contained on tape into a separate file. You would then create a new file.

2. Press **LOAD** (F2).

   The prompt `Load from:` will appear on the bottom of the Screen.

3. Type the file name of the data or program cassette file and press ENTER.

   You will hear a high-pitched sound which indicates that the Computer is searching the tape for the file name you specified.

Once the file name has been located, the prompt `Load from:` will change to:

`FOUND:` *filename*

where *filename* is the name of the file you specified.

If the cassette contains several files or programs, the Model 100 will skip over them until the desired one is found. You will know this is happening because every time an undesired file name is encountered the message:

`SKIP:`*filename*

will appear on the bottom of the Display.

Whenever you load a file from a cassette into a Computer file that already has some text in it, the loaded information will be "tacked" onto the end of the file. You can then move the newly loaded information to a different location in the file using the Select, Cut, and Copy functions.

Using the SAVE Function

If you have a Text file you wish to save onto cassette tape, use the **SAVE** (F3) function.

To save TEXT files or programs onto tape:

Be sure the Model 100 is properly connected to a suitable cassette recorder (see Appendix A).

1. Access or create the Text file you wish to save on tape.
2. Press **SAVE** (F3). The prompt `Save to:` will appear at the bottom of the Screen.
3. Type a file name (no longer than six characters in length) of the file which contains the information stored on tape and press ENTER.

Once the file has been saved on tape, the prompt will disappear from the Screen.

Using Control Codes and Other Special Key Combinations

The CTRL Key, used in conjunction with other keys, allows you to perform many special operations when the Model 100 is executing the TEXT program. These include:

- Cursor movement of one space, one word, or one line.
- Saving a file to cassette without entering the SAVE command.
- Printing a file.
- Cut and Paste

and more.

As you can see, these control code combinations duplicate the actions of the Function Keys, Command Keys, and Cursor Movement Keys. Note that it is not necessary for you to be familiar with the control code sequences to use TEXT. However, if you do become familiar with the control codes, your fingers will never have to leave the main keyboard to use TEXT to its full capacity.

To use a Control Code key-sequence, press CTRL and the predefined key simultaneously. Table 8-3 describes the operations performed by the various Control Code sequences.

Control Code	Operation Performed
CTRL A	Moves the Cursor from current position to the beginning of the first word to the left.
CTRL B	Moves the Cursor from current position to the bottom of the Display.
CTRL C	Cancels a SELECT, SAVE, LOAD, FIND, or PRINT function.
CTRL D	Moves the Cursor one character to the right.
CTRL E	Moves the Cursor up one line from current line.
CTRL F	Moves the Cursor to the beginning of the next word to the right.
CTRL G	Saves a file or a program.
CTRL H	Deletes previous character.
CTRL I	TAB.
CTRL L	Same as SELECT Function Key.
CTRL M	Carriage Return and Line Feed.
CTRL N	Same as FIND Function Key.
CTRL O	Same as COPY Function Key.
CTRL Q	Moves the Cursor to the left-most position of the current line.
CTRL R	Moves the Cursor to the right-most position of the current line.
CTRL S	Moves the Cursor one character to the left.
CTRL T	Moves the Cursor to the top of the Display in the current column.
CTRL U	Same as CUT Function Key.
CTRL V	Same as LOAD Function Key.
CTRL W	Moves the Cursor to the beginning of the current file.
CTRL X	Moves the Cursor down one line from its current position.
CTRL Y	Prints the entire file.
CTRL Z	Moves the Cursor to the end of the current file.

Table 8-3

CTRL P will allow you to ''embed'' printer codes (for boldface, underlining, etc.) in a Text file. These files must then be printed using the general device command SAVE TO: LPT: (press F3 and type LPT:). Even though the codes will appear on the Screen, they will not be printed on the printer. (If you print the file using SHIFT PRINT, the codes will be ignored and printed out on the printer.) For instance, to underline on the Daisy Wheel II printer, type CTRL P CTRL O. (CTRL O sends a decimal 15 which will start underlining.) The Screen will display ^O. To end underlining, move the Cursor and type CTRL P CTRL N. (CTRL N sends a decimal 14 which will stop underlining.) The Screen will display ^N. Then type F3 LPT: ENTER. SAVE LPT: does not use the WIDTH feature; you will have to format your text and place carriage returns where needed.

9 / Schedule Organizer (SCHEDL)

The Schedule Organizer program (SCHEDL) keeps track of and locates dates, and times for appointments or events. It also keeps records of information such as expense account items.

To use SCHEDL, you must first use TEXT to create a file called **NOTE**. (The program automatically adds the extension .DO.) It's important to note that there can be only one **NOTE.DO** file listed on the Main Menu.

In the NOTE file, you may store various records, each consisting of a date, a time, a place, and a note to yourself. For instance:

06/17 1:30 Staff meeting. South conference room.

This is only a suggested format. The format you should choose is the one which best suits your needs.

If you attempt to access the SCHEDL Application Program without creating file NOTE, the message:

```
NOTE.DO not found
Press space bar for MENU
```

will appear on the Display and a beeper will sound. To return to the Main Menu, press **SPACEBAR**.

Using the Function Keys in SCHEDL

The Function Keys **F1** - **F8** have unique definitions when you're using the SCHEDL program.

The definitions for the Function Keys are immediately displayed on the bottom of the Screen and will look like Figure 9-1.

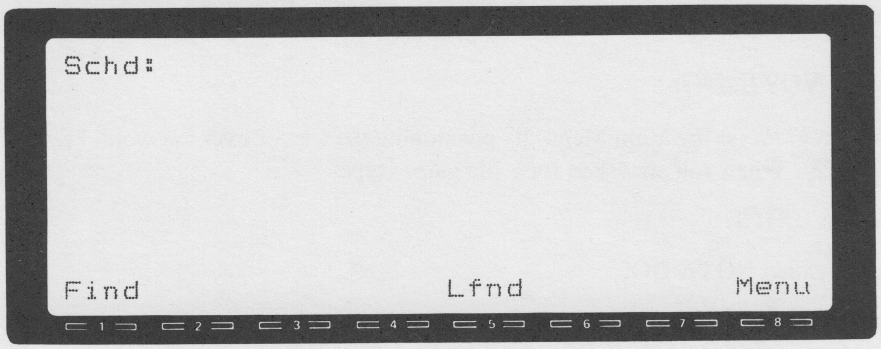


Figure 9-1. SCHEDL Function Key Definitions

Pressing **LABEL** causes the definitions to disappear from the Screen.

Find Pressing (F1) allows you to "find" a specified record from the NOTE file. Press (F1) and type in the item (letters or numbers) you want to look for. The Cursor will move to the first occurrence of what you specify. This chapter will provide details on using this function.

(F2) is not used by the SCHEDL Program.

(F3) is not used by the SCHEDL Program.

(F4) is not used by the SCHEDL Program.

Lfnd (F5) works exactly like (F1) except the information is listed on the printer (if one is connected) instead of the Display. All "matched" items will automatically be printed without pausing.

(F6) is not used by the SCHEDL Program.

(F7) is not used by the SCHEDL Program.

Menu Press (F8) to exit the SCHEDL Program and return to the Main Menu.

To cancel any operation, press (BREAK) ((SHIFT)(PAUSE)).

Using the Command Keys in SCHEDL

The Command Keys have the same definition in all of the Application Programs. See Table 9-1.

Command Key Definitions	
Key	**Operation**
(PASTE)	Allows you to insert ("PASTE" in) text that has previously been COPYed or CUT.
(LABEL)	Displays the current definitions of the Function Keys.
(PRINT)	Prints on a printer whatever is currently on the Display.
(BREAK)	To cancel any operation, press (BREAK) ((SHIFT)(PAUSE)).

Table 9-1

To create the file NOTE.DO:

1. Enter the TEXT program (at the Main Menu) by positioning the Cursor over the word `TEXT` and pressing (ENTER). When you are asked for a file name, type:

   `NOTE` and press (ENTER).

   This creates a file called **NOTE.DO**.

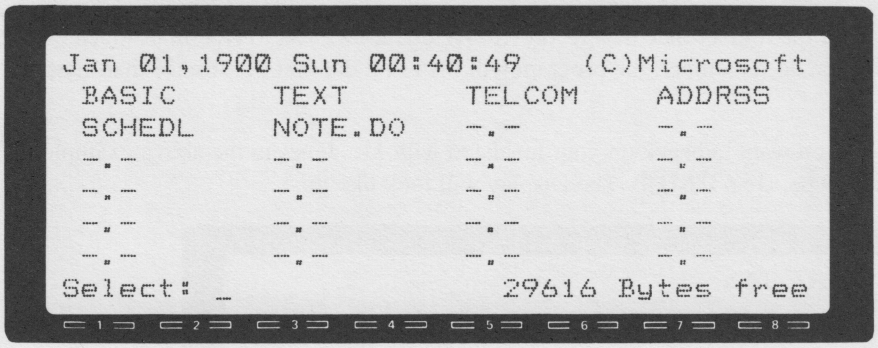
2. The Display will clear, leaving the Cursor in the upper-left corner. You have now created file NOTE and may begin typing in information ("records") such as dates and times for meetings. End each entry (or record) by pressing (ENTER).

For instance, type the following information in the NOTE file:

```
9/18 Product Schedules due (ENTER)
9/23 2:30 Staff meeting (ENTER)
10/3 11:45 Business luncheon with Jones (ENTER)
```

3. Exit TEXT by pressing (F8).

When you do, the Main Menu will look like Figure 9-2.



```
Jan 01,1900 Sun 00:40:49   (C)Microsoft
 BASIC      TEXT       TELCOM     ADDRSS
 SCHEDL     NOTE.DO    -.-        -.-
 -.-        -.-        -.-        -.-
 -.-        -.-        -.-        -.-
 -.-        -.-        -.-        -.-
 -.-        -.-        -.-        -.-
Select: _                29616 Bytes free
```

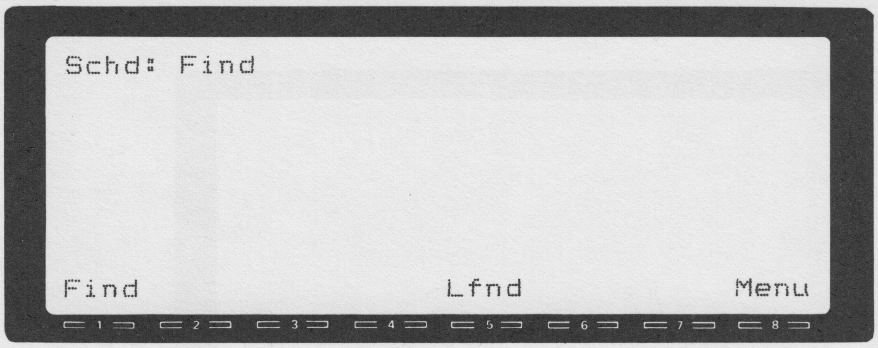
Figure 9-2. NOTE.DO File on the Main Menu

Using SCHEDL to retrieve information from the NOTE file . . .

1. To access and use the file NOTE.DO from the Scheduler Organizer program, enter the Scheduler Program from the Main Menu, by positioning the Menu cursor over the word `SCHEDL` and pressing (ENTER). You will be prompted:

```
Schd:
```

2. To locate any record from the NOTE file, press (F1). The Screen will display:



```
Schd: Find







Find                 Lfnd             Menu
```

3. Now type in the item you wish to find and press (ENTER). That record containing the specified item will then be displayed.

For example, to review your schedule for the 18th, type:

```
18 (ENTER)
```

since this occurs only once in the NOTE.DO file.

The Computer will search the NOTE.DO file for any occurrences the record you specified — regardless of its position within a line. In this case it will display:

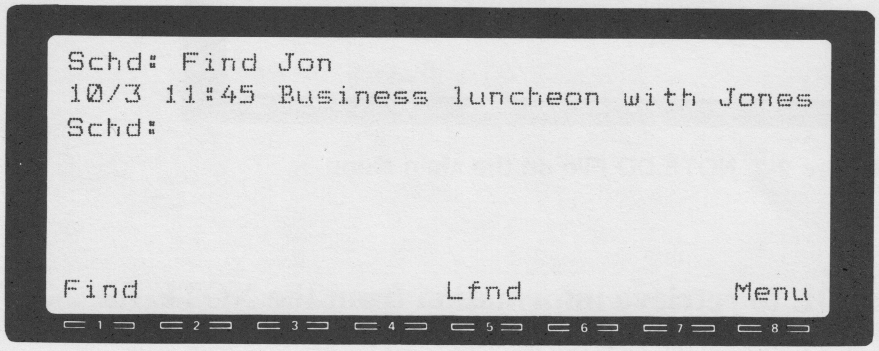
```
9/18 Product Schedules due
```

4. The prompt `Schd:` will reappear. At this time, you may press **FIND** (F1) again and specify a new item from the file.

   For instance, if you press **FIND** and type `23` (ENTER), the information concerning the staff meeting will be displayed.

5. If you can't recall the time or date of a record, SCHEDL allows you to type in as much of the information as you do remember; the program will then find the first occurrence that matches it.

   For example, if you want to check on your luncheon with Mr. Jones in the above example, press **FIND** and type: `Jon` (ENTER). The Display will look like this:

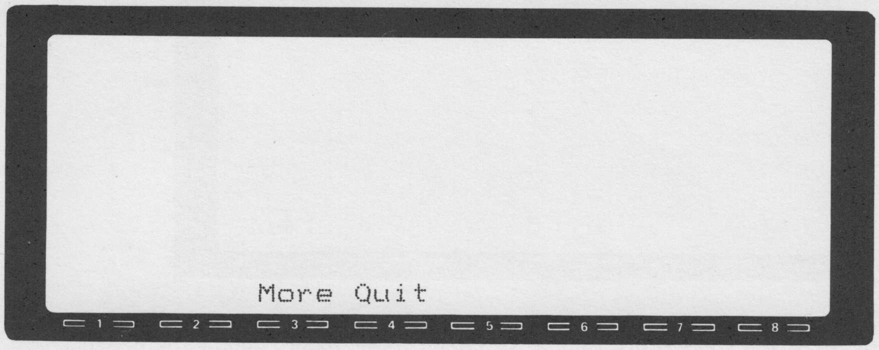


```
Schd: Find Jon
10/3 11:45 Business luncheon with Jones
Schd:




Find                    Lfnd               Menu
```

However, if the item you specify appears more than once in the NOTE.DO file, all the items in which it occurs will be displayed.

The More and Quit Prompts

When the records containing the specified items exceeds six lines (or the Display), the prompts:



```
             More Quit
```

will appear at the bottom of the Display.

- Press **MORE** ((F3) or (M)) if you want to see the next occurrence of the item you specified.

or

- Press **QUIT** ((F4) or (Q)) if you waht to search for a different item.

To return to the Menu, press (F8) when the prompt `Schd:` appears on the Screen.

Scanning the NOTE File

It is also possible to ''scan'' the NOTE file. When scanning, you may view six lines from the NOTE file at a time.

To scan the NOTE file:

1. Access the SCHEDL Application Program.
2. Press **FIND** ((F1)) and (ENTER).

Instantly the Display fills up with the first six lines of the NOTE file. The last line displays the message

```
More   Quit
```

If you wish to continue scanning press **MORE** ((F3)) and the next six lines will be displayed. If not, press **QUIT** ((F4)).

Sample SCHEDL Session

The real usefulness of SCHEDL stems from its ability to search for particular characters. This enables you to code and categorize the items in the NOTE.DO file. By doing so, it is then possible to retrieve related information selectively.

The following example illustrates the usefulness of the SCHEDL Application Program:

Suppose that you hold an executive position within a large firm. Your company is about to expand its line of products and for the next two months you face a very demanding schedule. (Assume the months are October and November.) You have to attend a national conference in Denver, Colorado where the details will be fully disclosed.

During your trip you must keep a record of your expense account items. In addition, you must meet with your staff to discuss the implications of the expansion on your department's activities.

On your return, you have set a dinner appointment to finalize a transaction with Jones, from the media department. At the same time, you have to prepare a budget proposal for the next quarter, and review the current advertising promotions.

Also, you must delegate the responsibility of updating the department's filing system to someone. As if all this wasn't enough, Amy, your daughter is getting married.

While you could rely on a calendar and an agenda to guide you through the maze that awaits you, the SCHEDL Application Program can simplify matters enormously by allowing you to extract information selectively.

Consider the tasks listed above:

- There are various appointments you must keep and certain things which you must do.
- You have to maintain a record of your expense account items while travelling to Denver.

To organize this information in a coherent manner, you can start by separating the information into ''appointments,'' ''must do's,'' and ''expense account items,'' and assign the symbols ''#,'' ''*,'' and ''$'' to each respectively.

Note: These symbols were randomly chosen. You can use any symbols or labels as long as they mean something to you and are used consistently within each group.

Let's categorize the appointments in this way:

10/22 1:00 National Conference, Denver, Colorado
11/3 3:15 Staff Meeting
11/3 4:30 Dinner with Jones
11/10 7:30 Wedding (Dallas) 100 Park Row

We can categorize the "must do's" in the same fashion, using, of course, the asterisk symbol:

* 10/26 Quarterly budget proposal due
* 11/5 Advertising Promotions Report due
* 11/8 See Ann about file system update

Finally, we can categorize the items you placed on your expense account using the dollar sign symbol.

$ 10/17 270.00 Plane tickets (Denver trip)
$ 10/21 97.00 Hotel (Denver)
$ 10/22 25.00 Taxi/tips (Denver trip)
$ 10/23 13.00 Airport Parking

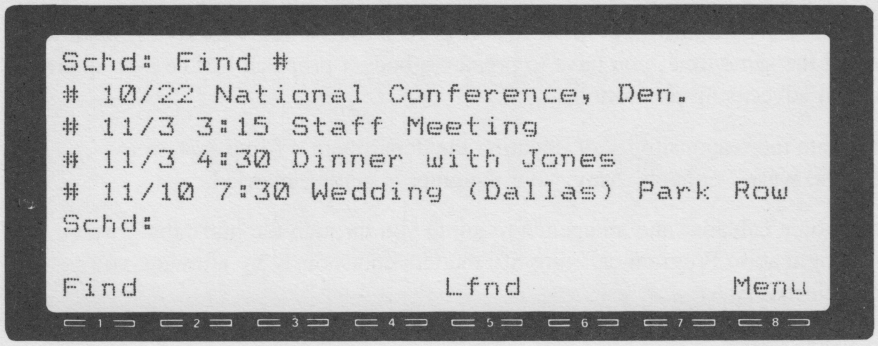
and so on.

Once this information has been entered into the NOTE file with the appropriate coding symbols preceding them, it is possible to retrieve groups of related information selectively.

For instance, if you wish to be reminded of the appointments:

1. Access the SCHEDL Application Program by positioning the Cursor over `SCHEDL` and pressing **ENTER** (or by typing `SCHEDL` **ENTER**).
2. Press **FIND** (**F1**).
3. Type the symbol # and press **ENTER**.

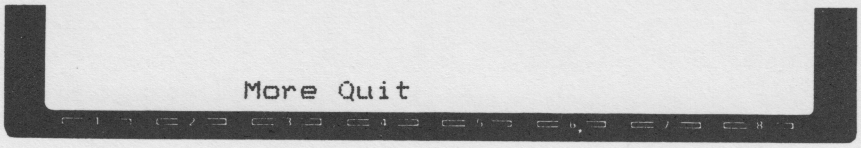
The Display will immediately show the items you previously entered in the NOTE file which contain #.



```
Schd: Find #
# 10/22 National Conference, Den.
# 11/3 3:15 Staff Meeting
# 11/3 4:30 Dinner with Jones
# 11/10 7:30 Wedding (Dallas) Park Row
Schd:

Find                 Lfnd               Menu
```

Next, press **FIND** (**F1**) and type either the asterisk symbol (*) or the dollar sign ($). This will display meetings or expense account items.

Whenever the number of records with the same identifying symbol exceeds six lines in length, the prompts More and Quit appear at the bottom of the Screen.



- To display the remaining items press **MORE** ((F3)).
- To display different items press **QUIT** ((F4)) and type the new coding symbol (*****, in the case of ''must do's,'' and **$**, in the case of expense account items).

When you wish to retrieve information from the NOTE file concerning only a certain month, a day, or a particular time it is only necessary to specify an item that makes the request unique.

Notice that the information from the previous example indicated month, day and time (the ''must do's'' did not include time). This was included to allow you greater flexibility in narrowing the information to be retrieved. For instance, if you wish to know what appointments you must keep in the month of September only:

Enter the SCHEDL Application Program, press **FIND**, and type:

```
# 10 ENTER
```

The Computer will search the NOTE file for **# 10** and display all the records in which it occurs. You can narrow the request down even further by specifying day or even a time. For instance, if you were to type:

```
# 11/3 3:15 ENTER
```

After pressing **FIND** ((F1)), the Computer would search and display only the record in which this occurs — the staff meeting. The other item in which at least part of this string occurs — dinner with Jones — would be ignored since you specified a time in the request.

10 / The Address Organizer (ADDRSS)

The Address Organizer Application Program lets you retrieve information (names, telephone numbers, and addresses) from the ADRS file. Furthermore, the information contained in the ADRS file can be used by the Telecommunications Program (TELCOM, see Chapter 11) to automatically dial telephone numbers.

To use ADDRSS, you must first use TEXT to create a file called **ADRS**. (The program automatically adds the extension .DO.)

It is important to note that there can only be one ADRS.DO file listed on the Main Menu.

If you attempt to access the ADDRSS Application Program without first creating the ADRS.DO file, the message:

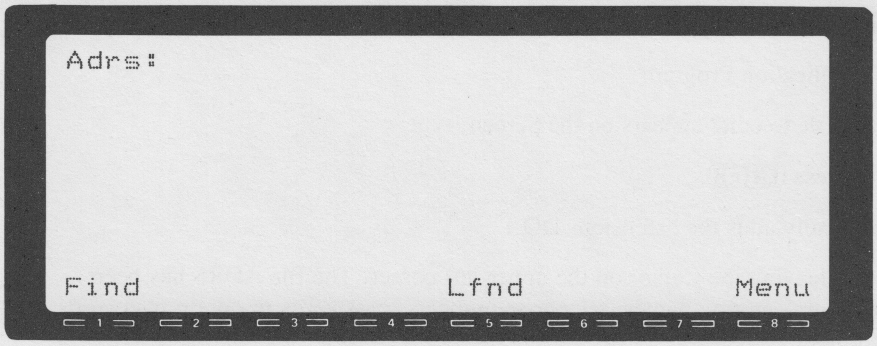
```
ADRS.DO not found
Press space bar for MENU
```

will appear on the Display and a beeper will sound. To get back to the Menu, simply press SPACEBAR.

Using the Function Keys in ADDRSS

The Function Keys F1 - F8 have unique definitions when you're using the ADDRSS program.

These unique definitions appear on the bottom of the Display when you access ADDRSS and will look like Figure 10-1.



```
Adrs:
```

Figure 10-1. ADDRSS Function Key Definitions

Pressing LABEL causes the definitions to disappear from the Display.

Find Pressing F1 allows you to "find" a specified record from the ADRS file. Press F1 and type in the item (letters or numbers) you want to look for. The Cursor will move to the first occurrence of what you specify. This chapter will provide details on using this function.

F2 is not used by the ADDRSS Program.

F3 is not used by the ADDRSS Program.

F4 is not used by the ADDRSS Program.

Lfnd F5 works exactly like F1 except the information is listed on the printer (if one is connected) instead of the Display.

F6 is not used by the ADDRSS Program.

F7 is not used by the ADDRSS Program.

Menu (F8) Press F8 to exit the ADDRSS Program and return to the Main Menu.

To cancel any operation, press BREAK (SHIFT PAUSE).

Using the Command Keys in ADDRSS

The Command Keys have the same definition in all of the Application Programs. See Table 10-1.

Command Key Definitions

Key	Operation
PASTE	Allows you to insert ("PASTE" in) text that has previously been COPYed or CUT.
LABEL	Displays the current definitions of the Function Keys.
PRINT	Prints on a printer whatever is currently on the Display.
BREAK	To cancel any operation, press BREAK (SHIFT PAUSE).

Table 10-1

To create the ADRS file:

1. Enter the TEXT Application Program.
2. When the message File to edit? appears on the Screen, type:

   `ADRS` and press ENTER.

(The program automatically adds the extension .DO.)

The Display will clear, leaving the Cursor on the upper-left corner. The file ADRS has been created and you may begin entering names, telephone numbers, passwords (more on passwords in Chapter 11), or addresses.

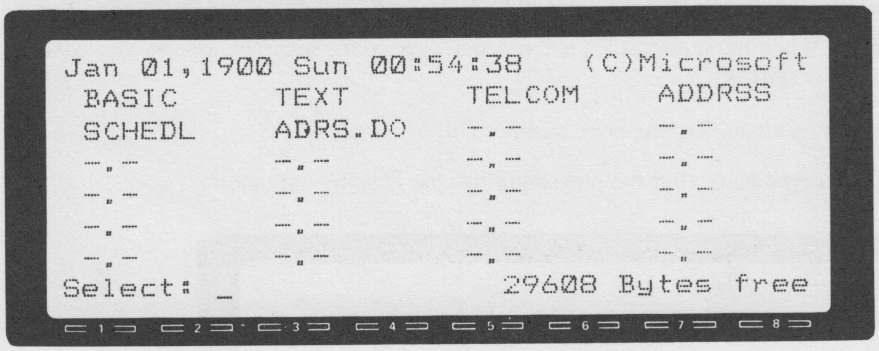
Before you begin entering information into the file, there are two important rules you should always remember!

- Include an optional "dash" in a telephone number between area codes and prefixes.

- *Always* put a colon before a telephone number. For instance, **:4179257973**. This will allow you to use the Auto-Dialing feature later on. A second colon after the phone number may be used to terminate dialing at that point if necessary.
- *Always* end an address entry by pressing ENTER. For instance:

```
Rick Schell :4179257973: 453 Red River ENTER
```

Return to the Main Menu by pressing **MENU** (F8). The file just created will be listed as ADRS.DO. (The program supplies the extension .DO.)



```
Jan 01,1900 Sun 00:54:38    (C)Microsoft
 BASIC       TEXT        TELCOM      ADDRSS
 SCHEDL      ADRS.DO     -.-         -.-
 -.-         -.-         -.-         -.-
 -.-         -.-         -.-         -.-
 -.-         -.-         -.-         -.-
 -.-         -.-         -.-         -.-
Select: _                 29608 Bytes free
```

Retrieving Information from the ADRS File

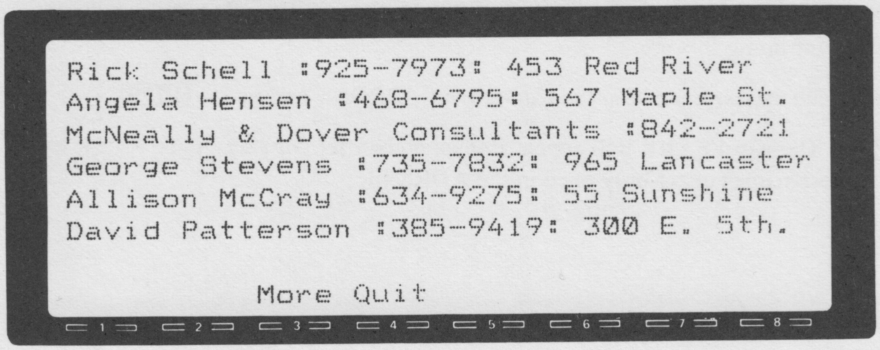
Once the ADRS file has been created, you can examine the names and addresses in two ways:

- Look at every name and address in general as they were created.
- Look at a particular address by pressing **FIND** (F1) and specifying the name, address, or telephone number you wish to see.

To examine the ADRS file in general, follow this procedure:

1. When the Main Menu appears, position the Cursor over `ADDRSS` and press ENTER.
2. The prompt `Adrs:` will appear. Press **FIND** (F1) and ENTER.

   The Screen will display six lines of addresses followed by the prompts `More    Quit`



```
Rick Schell :925-7973: 453 Red River
Angela Hensen :468-6795: 567 Maple St.
McNeally & Dover Consultants :842-2721
George Stevens :735-7832: 965 Lancaster
Allison McCray :634-9275: 55 Sunshine
David Patterson :385-9419: 300 E. 5th.

          More Quit
```

(If you limit addresses to a single line each, there will be six addresses displayed on the Screen. If there are less than six names in the file, the More and Quit prompts will not appear.)

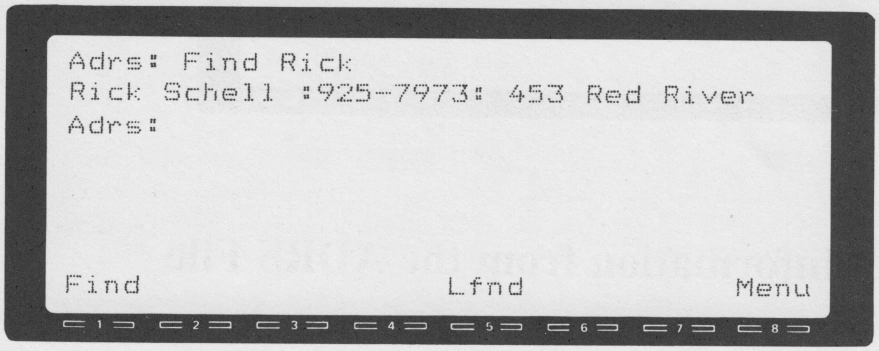
3. If the name or address you want to examine is not displayed, press **MORE** (F3) to display six more lines of information.

To examine a specific address in the ADRS file:

1. When the Main Menu appears, position the Cursor over `ADDRSS` and press ENTER.
2. Press **FIND** (F1). When the message `Find` appears, type in the name or address you wish to examine and press ENTER.

   The specified name, address, or telephone number will be displayed.

   For instance, if you type Rick after the pressing Find, the Display will show:



```
Adrs: Find Rick
Rick Schell :925-7973: 453 Red River
Adrs:



Find                   Lfnd                 Menu
```

ADDRSS Sample Session #1

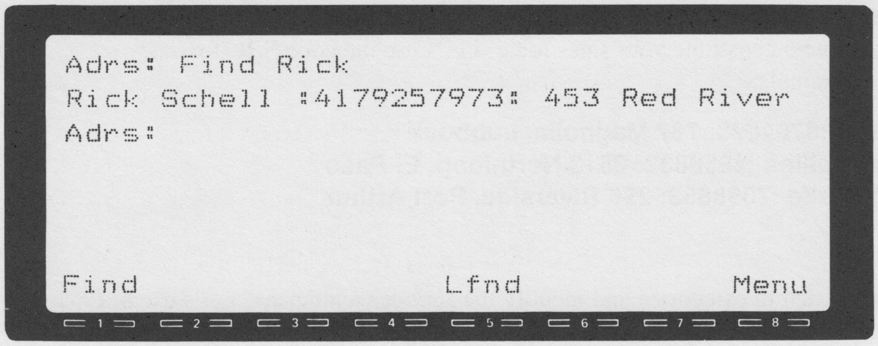
In this section, we will describe a sample ADDRSS session. To begin with, create the ADRS file using TEXT and store the following information:

Rick Schell :4179257973: 453 Red River
Angela Hensen :4686795: 567 Maple St.
McNeally & Dover Consultants :13398422721:

Return to the Main Menu by pressing F8.

Now, suppose you would like to retrieve Mr. Schell's address and phone number from the ADRS file. The steps for doing this are:

1. When the Main Menu appears, position the Cursor over `ADDRSS` and press ENTER.
2. Press **FIND** (F1). The word `Find` will appear immediately after `Adrs:`. Type `Rick` or `Schell` since these names occur only once in the ADRS file.



```
Adrs: Find Rick
Rick Schell :4179257973: 453 Red River
Adrs:




Find                Lfnd               Menu
```

3. Press ENTER.

The Model 100 will search the ADRS file for any occurrences of the name you specified — regardless of its position within a line — and display the entire line. In this case, Mr. Schell's address and phone number will be displayed:

```
Rick Schell :4179257973: 453 Red River
```

The prompt `Adrs:` will then appear. At this time, you may press **FIND** and type a different name.

For instance, if you type **Ang** or **Hen** after pressing F1, the information on **Ms. Hensen** will be displayed. On the other hand, by typing **McN** or **Dov**, you can display the information on **McNeally & Dover**. This procedure applies to retrieving any particular item from the ADRS file.

If the item you specify appears more than once in the ADRS file, the Screen will display all the items in which it occurs.

However, when the items containing this string exceed the area of display (six lines) the prompts **More** and **Quit** will appear at the bottom of the Screen. Press F3 if you wish to retrieve the next occurrence of this string, or press F4 if you want to search for a different string. To return to the Main Menu, press F8 when the prompt `Adrs:` appears on the Screen.

Sample Session #2

While the previous example demonstrated the basic operation of the ADDRSS Application Program, the Model 100 is much more than just an electronic phone book. The ability to search for occurrences of a particular item in the ADRS file opens a wide horizon of practical uses.

For instance, by coding the information in the ADRS file using different characters and symbols, you can retrieve related information selectively. The following example illustrates how this is done:

Assume you are a regional manager for a company that sells and services microcomputers to small businesses scattered throughout Texas.

You are in charge of a sales force consisting of eight representatives and, in addition, you also direct a large fleet of service technicians. It is important that you know at all times which companies each of your salespeople is assigned to and also which technicians are responsible for providing service.

We can begin categorizing this information by attaching an arbitrary symbol or character string to the names of those who constitute your sales force. Let's use the code **S.R** (for sales representative). For example:

S.R Al Bester :3764628: 767 Magnolia, Lubbock
S.R George Collins :4958832: 2513 Northloop, El Paso
S.R Ron Metcalfe :7358653: 256 Riverside, Port Arthur

And so on.

Another code may be used to categorize and identify the service technicians. Let's use **S.T.** (for service technician). For example:

S.T. Henry Ellison :2548164: 683 Sunshine Blvd, Lubbock
S.T. David Fuller :7731743: 948 West Maple, Houston
S.T. Tom Manning :4362946: 3287 Royal Lane Dallas

Finally, we can use another code for the various businesses that your company serves. Let's use the code **ACCT** (for account).

To determine the sales representatives and technicians associated with each account, we can simply attach their initials at the end of each business in question. For clarity, we may enclose the initials of the technician in parenthesis. For instance:

ACCT Ace Auto Parts Supply :7644736: 6392 Rundberg, Dallas Tx AB (DF)

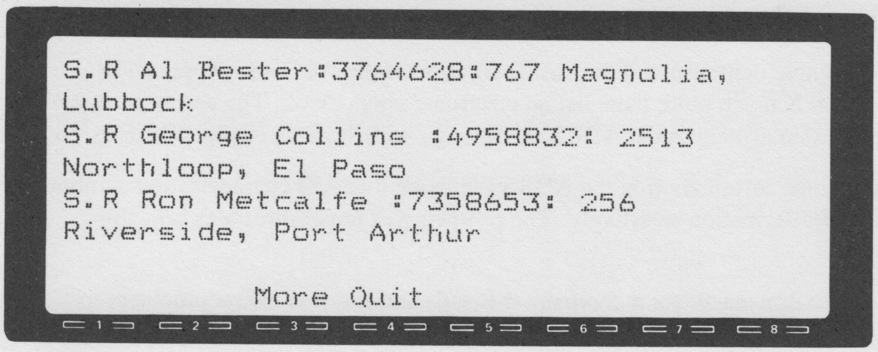
where **ACCT** is the code attached to every account and the initials at the end, **AB (DF)**, identify the sales representative, **Al Bester**, and the service technician, in parenthesis, **David Fuller**.

After following the same procedure with each account and storing all this in the ADRS file, it is then possible to retrieve groups of related information selectively.

For instance, to retrieve the names of the sales representatives:

1. When the Main Menu appears, position the Cursor over `ADDRSS` and press ENTER.
2. Press **FIND** (F1) and type the code `S.R` ENTER.

The Screen will display the items in the ADRS file which contain the code you specified.



```
S.R Al Bester:3764628:767 Magnolia,
Lubbock
S.R George Collins :4958832: 2513
Northloop, El Paso
S.R Ron Metcalfe :7358653: 256
Riverside, Port Arthur

           More Quit
```

Follow the same procedure to retrieve the names of the service technicians, and also those of the businesses. Substitute, of course, the code for S.T, and ACCT respectively.

11 / Computer-to-Computer Communication (TELCOM)

TELCOM allows your Model 100 to be used as an automatic telephone dialer and to communicate with other computers over the telephone lines. This includes other TRS-80 microcomputers as well as a variety of ''host'' computers systems and information services such as Dow Jones and CompuServe.

Information that is received from another computer can then be printed out (if a printer is connected) or stored in the Model 100's memory for later printing or viewing.

TELCOM has two ''modes'' of operation:

- **Entry Mode**, in which you use the Model 100 to dial (automatically or manually) a telephone number for either normal conversation (once you pick up the phone) or for computer-to-computer communications. When you enter TELCOM, you are automatically in Entry Mode.
- **Terminal Mode**, in which the Model 100 communicates with another computer system or information service.

Entry Mode When you start up the TELCOM program, you are automatically in Entry Mode which is ''stand alone.'' That is, you can use it as a feature in itself just as you would any other telephone automatic dialer. However, the auto-dialing function is available only when the Model 100 is connected directly to a modular telephone line via the optional/extra ***Model 100 Modem Cable*** (26-1410).

Terminal Mode, is used for interactive information exchange with another computer system. When you use the two operation modes together, not only can the Model 100 automatically dial an information service number, but it can also automatically ''log-on'' to the system.

It's important to note that the Function Keys F1 through F8 have different definitions and functions in these two operation modes.

To perform operations such as auto-dialing, TELCOM uses the names and telephone numbers that you have previously stored in the ADRS.DO (see Chapter 10 of this manual).

To use TELCOM, the Model 100 must be connected to a telephone or telephone line in one of several ways. This includes:

- The Model 100's built-in modem and the ***Model 100 Modem Cable*** (26-1410).
- The ***Model 100 Acoustic Coupler*** (26-3805).

If you are communicating directly with another computer (i.e., not over the telephone lines), a ***Null Modem Adapter*** (26-1496) and a Radio Shack ***RS-232C Cable*** (such as 26-1490) are used. Note that a ***Cable Extender*** (26-1495) may also be required in some cases.

One of the above connections (either directly or over the phone lines) must be made before the Model 100 can communicate with another computer. This chapter will describe how to make these connections in addition to describing how TELCOM works.

Connecting the Model 100 to the Telephone Lines

You can connect the Model 100 to a telephone line in two different ways:

- Directly using the optional/extra ***Model 100 Modem Cable*** (26-1410).
- Through an external acoustic coupler such as the optional/extra ***Model 100 Acoustic Coupler*** (26-3805).

If at all possible, we recommend you use the direct connect built-in modem which usually provides easier connections and more reliable information transfer.

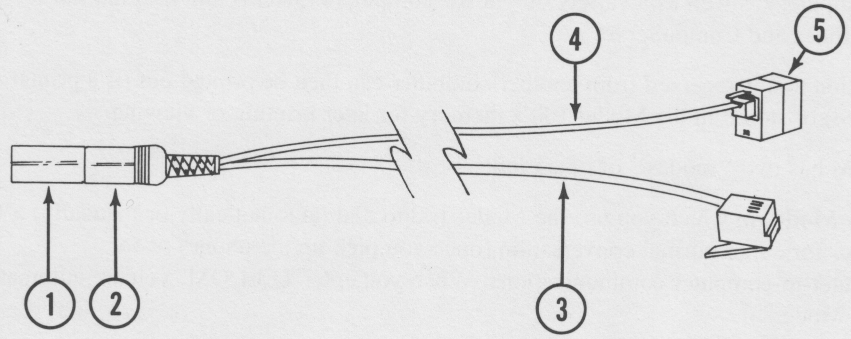


Figure 11-1. Model 100 Modem Cable (optional/extra, 26-1410)

① **Shorting Plug** Remove this plug to insert the round cable connector into the **PHONE** connector on the rear panel of the Model 100. Attach this plug to the cable when you disconnect the cable from the Computer but do not disconnect the cable from the phone.

② **Computer Connector** Insert this cable into the **PHONE** Connector on the Model 100.

③ **Silver Telephone Cable** Insert the connector on this cable into the telephone connector.

④ **Beige Telephone Cable**

⑤ **Telephone Connector** Connect the plug you remove from the telephone into this connector. This connects the cable to the telephone wall outlet.

Direct Connection

To connect the Model 100 directly to the telephone lines, you must use an optional *Modem Cable* (26-1410).

Note: The auto-dialing function is available only when the Model 100 is connected directly to a modular telephone line.

Before connecting your Model 100 to the phone lines, notify your local telephone company of:

Manufacturer: Radio Shack
Model: TRS-80 Model 100 Computer with Built-in Modem 26-3801
FCC ID: AWQ9SB26-3801
FCC Registration Number: AWQ9SB-70372-DT-R
Ringer Equivalence Number (REN): 0.0B

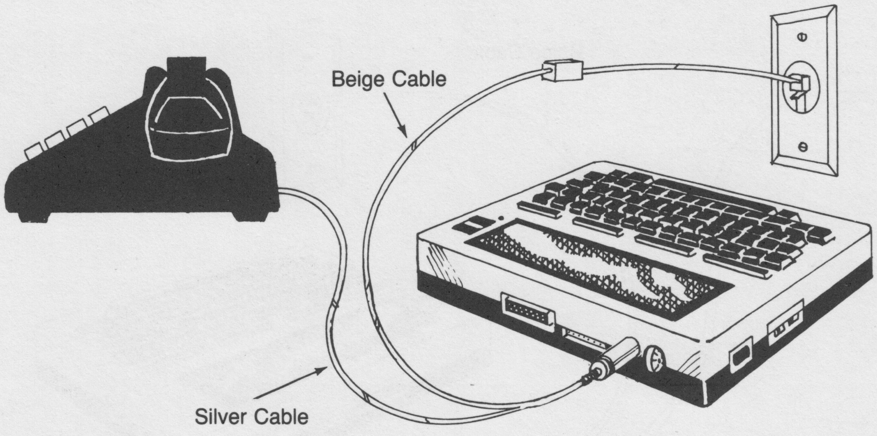


Figure 11-2. Model 100-to-Telephone Connection

To directly connect the Model 100 to a Telephone:

1. Disconnect the **Shorting Plug** from the round end of the Modem Cable.
2. Connect the round end of the Modem Cable to the **PHONE** connector on the rear panel of the Model 100.
3. Disconnect the telephone line from the phone — *not from the wall outlet!*
4. Insert the telephone line which is connected to the wall outlet into the Telephone Connector on the Modem Cable — the beige box at the end of the beige wire.
5. Connect the telephone plug from the Modem Cable — *the silver wire* — into the telephone.
6. Set the **DIR/ACP** Switch (on the left side of the Computer) to **DIR** (for Direct).

If you remove the Model 100 from this Direct Connection, you must replace the Shorting Plug onto the round end of the Modem Cable. This will allow you to use the telephone when the Modem Cable is no longer connected to the Model 100.

If you don't need to have the telephone connected . . .

Simply remove the beige box at the end of the beige cable and insert the connector of the beige cable directly into the wall outlet.

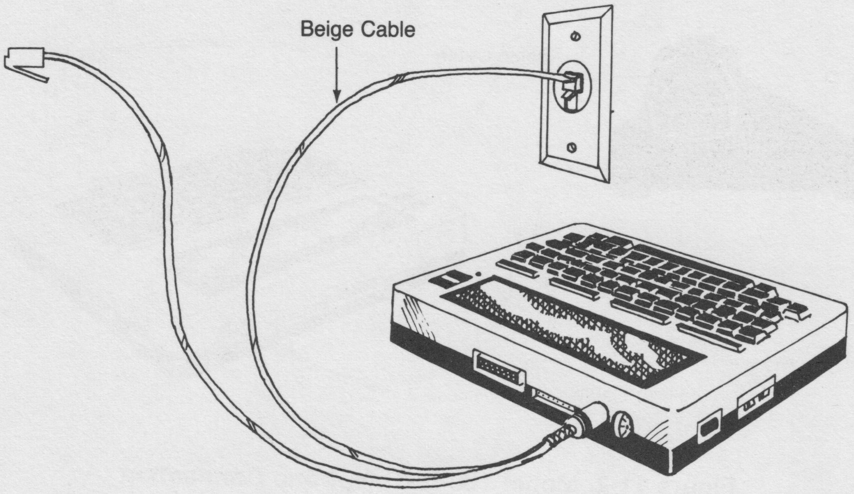


Figure 11-3. Model 100-to-Telephone Line Connection

Acoustic Coupler Connection

At times, it may be impossible to disconnect the telephone plug from the back of the telephone. This situation is often encountered in hotel rooms.

The optional/extra ***Acoustic Coupler*** (26-3805), which consists of a speaker and a microphone that attach easily to a telephone receiver, is suited for occassions such as these. However, the auto-dialing function is not available when the Acoustic Coupler is used.

To connect the Model 100 to an Acoustic Coupler:

1. Connect the round end of the Coupler Cable to the **PHONE** connector on the rear panel of the Computer.
2. Slip the Acoustic Coupler Speaker over the microphone of the telephone receiver.
3. Slip the Acoustic Coupler Microphone over the telephone Speaker.
4. Set the **DIR/ACP** Switch (on the side panel of the Model 100) to **ACP** (for Acoustic Coupler).

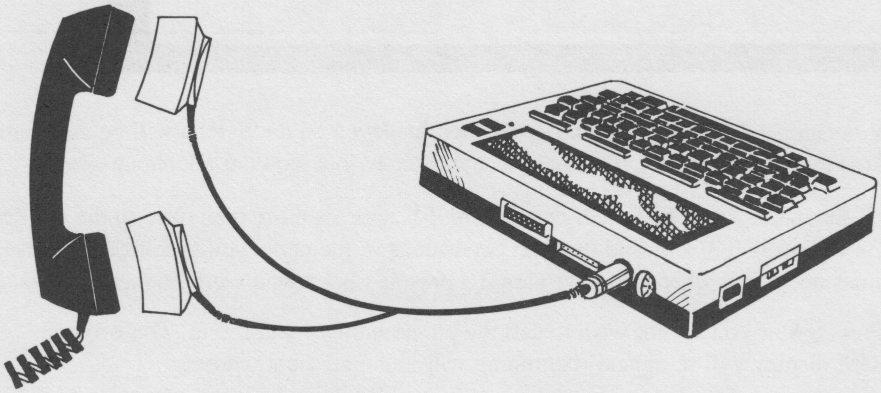
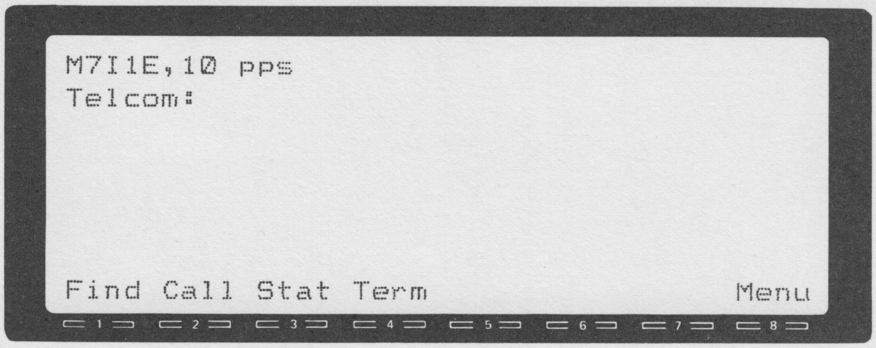


Figure 11-4. Connecting the Model 100 Acoustic Cups

Using the Function Keys in Entry Mode

In Entry Mode, the Function Keys let you perform many complex communication operations easily and efficiently. For your convenience, the definition of the Function Keys appears on the last line of the Display at all times in TELCOM. See Figure 11-5. Pressing (LABEL) will cause the bottom line to disappear.

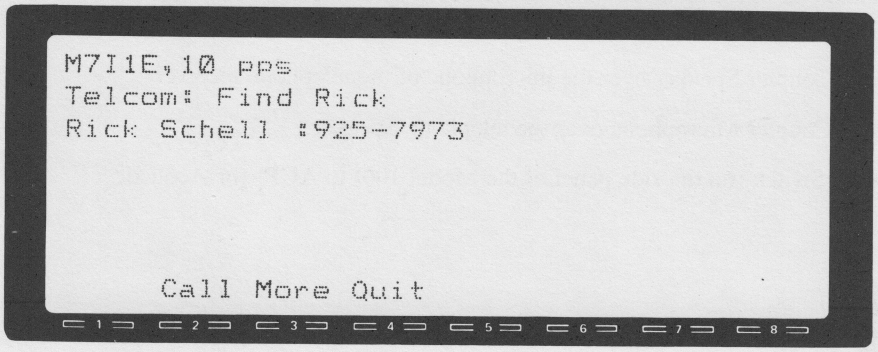


```
M7I1E,10 pps
Telcom:






Find Call Stat Term                Menu
```

Figure 11-5. TELCOM Function Key Definition

Find Pressing (F1) allows you to search for names and phone numbers that you stored in the ADRS.DO file. To use the Find function, press (F1), type a name or number (usually a name), and press (ENTER).

TELCOM will then display the name and phone number (up to the second colon in the record) and prompt you for the next action by displaying:



```
M7I1E,10 pps
Telcom: Find Rick
Rick Schell :925-7973



     Call More Quit
```

- **Call** You can press F2 and the phone number displayed on the Screen will be automatically dialed (as long as the Model 100 is connected directly to a modular telephone line).
- **More** If the name you specified after pressing F1 appears more than once in the ADRS.DO file, you can press F3 and find the next occurrence of the name you specified. This is sometimes necessary when you have stored a person's home and work phone numbers.
- **Quit** Press F4 if you do not wish to call the phone number you found. The original TELCOM display will re-appear (beginning with the `Telcom:` prompt).

If a match is not found, the prompt: `Telcom:` will re-appear.

Call Press F2, type the phone number you wish dialed, and press ENTER. The number you typed in will be dialed automatically. You will, however, need to pick up the receiver before auto-dialing is completed. (Remember that the Model 100 must be connected directly to modular telephone lines.)

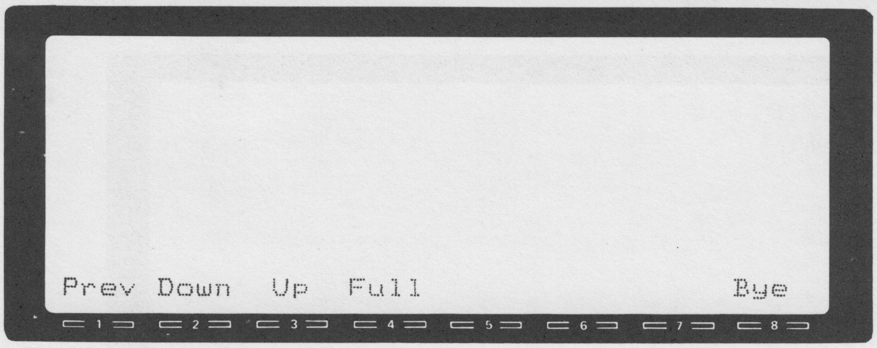
Stat Pressing F3 displays the current communications protocol (baud rate, word length, etc.) that TELCOM is using. Communications protocol between the Model 100 and the other computer must match.

Term After dialing a host system's telephone number, you can enter the Terminal Mode manually by pressing F4.

Menu Press F8 to exit TELCOM and return to the Main Menu.

Using the Function Keys in Terminal Mode

In Terminal Mode, the bottom line of the Display will change to remind you of the definition of Function Keys in this mode. Pressing LABEL will cause the bottom line to disappear.



Prev In Terminal Mode, the Display shows eight lines at a time. You may think of these eight lines as the bottom half of a page of text. To view the previous eight lines, it is only necessary to press F1. Pressing F1 again returns the last eight lines to the Screen.

Download Pressing F2 allows you to save incoming information in memory for viewing or printing later by creating a new file to store the information the host sends.

When you press F2, the prompt: `File to Download?` will appear. Type in a file name and press ENTER. To stop downloading, press F2 again.

Upload F3 allows you to send information that has been previously prepared in the TEXT Application Program to a host system.

When you press F3, the prompt: `File to Upload?` will appear. Type the name assigned to the file you wish to send to the host and press ENTER. The prompt: `Width:` will appear. Type in a number between 10 and 132 and press ENTER.

Full Pressing F4 lets you switch between Full and Half Duplex.

Most host systems require you to use Full Duplex. This means that any character you type is sent to the host before it appears on your Model 100 Display. If the characters you type are the same ones that appear on the Display, good communication with the host has been established.

Half Duplex, on the other hand, shows what you type directly on your Model 100 Display. This means that you have no way of knowing if the host received the same characters. (''Noisy'' telephone lines are sometimes the cause for this.)

Echo F5 enables you to obtain a printout (or ''hard copy'') of incoming information (if a printer is connected).

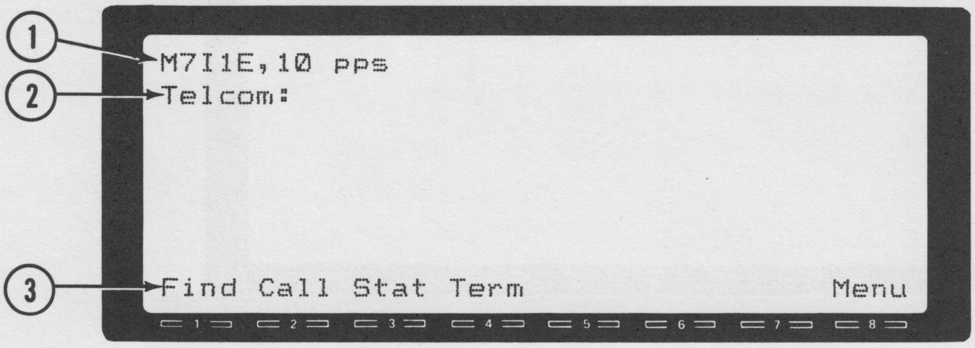
Bye Pressing F8 exits Terminal Mode and disconnects (or ''hangs up'') the telephone lines.

When you press F8, the prompt: `Disconnect?` will appear. At this time you may decide whether to terminate communcations by pressing Y (for yes) or N (for no) and press ENTER.

Important Note! Pressing **BYE** (F8) followed by Y is necessary for the Model 100 to ''release'' the telephone line.

Accessing TELCOM

You can access TELCOM from the Menu in the same way that you access other Application Programs. At the Main Menu, move the Cursor over the word `TELCOM` and press (ENTER). When you access TELCOM the display shows:



```
M7I1E,10 pps
Telcom:





Find Call Stat Term                 Menu
```

① The first line indicates the status of the communication parameters. This includes baud rate, word length, and parity.

② The second line shows the prompt `Telcom:` and lets you select one of the functions displayed on the last line of the Display.

③ The last line displays the definition of the Function Keys (F1) - (F8) in TELCOM.

Find Press (F1) to "search for" a specific phone number in the ADRS.DO file.

Call Press (F2) to dial a phone number.

Stat Pressing (F3) will let you know what the current status of the communication protocol.

Term Pressing (F4) manually puts you into Terminal Mode.

Menu Pressing (F8) returns you to the Main Menu.

Using Entry Mode

When you enter the TELCOM Program, you are automatically in Entry Mode. This allows you to dial phone numbers either "manually" (by typing the phone number on the Model 100 keyboard) or "automatically" (by letting the TELCOM program dial a number stored in the ADRS.DO file).

For details on manual and automatic dialing when you want to communicate with a host system, see **"Using Terminal Mode"** later in this chapter.

Manual Dialing

From the Keyboard . . .

It's possible to dial a telephone number without using the ADRS.DO file. To do this:

1. Access TELCOM (at the Main Menu) by positioning the Cursor over the word `TELCOM` and pressing ENTER.
2. Press **CALL** (F2).
3. When prompted, type the phone number you want to dial.
4. Press ENTER.

The message `Calling` will appear on the Display, followed by the phone number digits as they are auto-dialed.

Be sure to lift up the telephone receiver before dialing is complete.

From the Telephone . . .

When the Model 100 is connected to a telephone, it's also possible to dial a number in the conventional way by lifting the receiver and dialing the phone number.

Automatic Dialing

TELCOM lets the Model 100 automatically dial any phone number that's in the ADRS.DO file. For details on creating ADRS.DO files, see Chapter 10.

Remember that auto-dialing is only available when the Computer is directly connected to modular telephone lines using the Model 100 Modem Cable.

Be sure to lift up the telephone receiver sometime before the auto-dialing process has been completed.

To see how the auto-dialing feature works, type the following information into the ADRS.DO file:

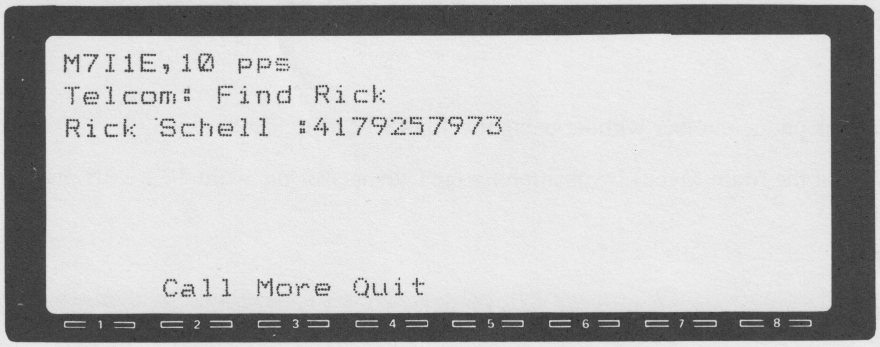
```
Rick Schell :4179257973: 453 Red River ENTER
Angela Hensen :4686795: 567 Maple St. ENTER
McNeally & Dover Consultants :13398422721: ENTER
```

Assume you wish to call Mr. Schell using the auto-dialing function:

1. Access TELCOM by positioning the Cursor over `TELCOM` and pressing ENTER.
2. Press **FIND** (F1).
3. When prompted, type the part of name that will make the request for Richard's telephone unique and press ENTER.

   For instance, type `Ric` or `Sch`. (You may use either upper- or lowercase letters.)
4. Press ENTER.

   When you do this, Rick's name, followed by his telephone number, will be displayed.



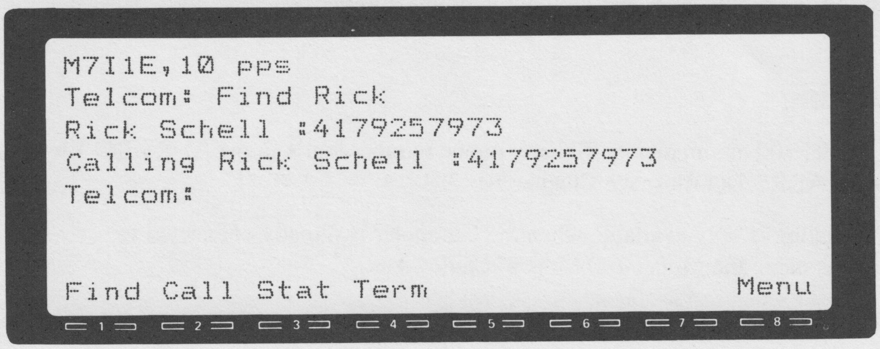
```
M7I1E,10 pps
Telcom: Find Rick
Rick Schell :4179257973




     Call More Quit
```

Notice the prompts: **Call**, **More**, and **Quit** at the bottom of the Display.

5. Press **CALL** (F2). The Screen will immediately display the message:

   `Calling Richard Schell:`

   The numbers which make up the entire phone number will appear on the Display one-by-one as they are dialed.



```
M7I1E,10 pps
Telcom: Find Rick
Rick Schell :4179257973
Calling Rick Schell :4179257973
Telcom:


Find Call Stat Term             Menu
```

Note: Remember to lift the receiver before auto-dialing is complete.

If you choose not to call the person whose name appears on the Display, press **QUIT** (F4) or Q.

Pressing either key returns the `Telcom:` prompt to the Screen. You can then "find" and "call" a different person or return to the Main Menu by pressing F8.

When a person's name appears more than once in the ADRS.DO file, press **MORE** (F3), or M to display its next occurrence.

Using Terminal Mode

Terminal Mode allows the Model 100 to communicate with a variety of host computers and information services.

When the Model 100 is linked either directly or over the telephone to a host, TELCOM's Terminal Mode — with its upload and download features — enables you to send or receive information.

You can even print information as it is being received from a host by pressing F5 (the "Echo" Function Key) or save that information in memory for viewing or printing at a later time.

The dialing procedure for Terminal Mode is slightly different from the dialing procedure if you're using the Model 100 strictly as an automatic dialer.

For one thing, you must be sure the ANS/ORIG Switch (on the left side of the Computer) is set to **ORIG**. You must also be sure the communication parameters (baud rate, etc.) for both your Model 100 and the host system match.

Communication Parameters

The Model 100's communication parameters (a series of conventions for transmitting and receiving information) must be set to match those of the computer on the ''other end'' before communications can take place.

That is, if you are sending information to another computer at 300 ''baud,'' it must be set at 300 ''baud'' to receive that information.

Table 11-5 at the end of this chapter defines the communication parameters. Remember that it isn't necessary to understand all there is to know about data communication (which can be a complicated subject) to use the features of Model 100 TELCOM.

To make it easy, TELCOM automatically sets the communication parameters for you. If your host system matches the settings described in Table 11-1, you shouldn't have to make any changes to communication parameters at all. See your host system user's guide for details on its communication parameters.

TELCOM Start-Up Communications Parameters

Meaning	Start-Up Setting
Baud Rate	M (300 baud)
Word Length	7 (bits)
Parity	I (Ignore Parity)
Stop Bit	1
Status	E (Enable)
Dial Pulse Rate	10 pps

Table 11-1

If you do need to change communication parameters, Table 11-2 describes the allowable settings.

Model 100 Telecommunications Protocol

	You Type:	For:
Baud Rate	M	"modem" (300)
	1	75 baud *
	2	110 baud
	3	300 baud
	4	600 baud
	5	1200 baud
	6	2400 baud
	7	4800 baud
	8	9600 baud
	9	19200 baud *
Word Length	6	6 bits
	[✓ 7]	7 bits
	8	8 bits
Parity	I	Ignore parity
	O	Odd parity
	[✓ E]	Even parity
	N	No parity
Stop Bit	[✓ 1]	1 stop bit
	2	2 stop bit
Line Status**	E	Enable (XON)
	D	Disable (XOFF)
Pulse Rate	10	10pps
	20	20pps

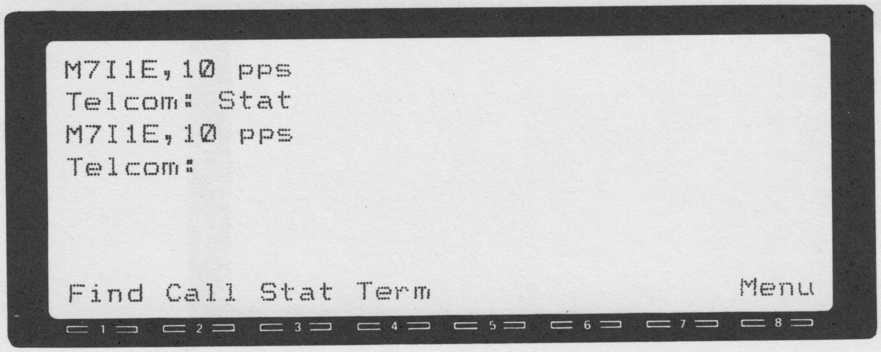
Table 11-2

*** Note:** The Model 100 uses 300 baud when the built-in modem is in use. If you use a number to set the baud rate, even if that number is 3 (for 300 baud), the modem becomes disabled. The RS-232C Interface then becomes enabled. For this reason, always select the letter M whenever the built-in modem is to be used.

**To ''manually'' send an XON, type CTRL Q. To ''manually'' send an XOFF, type CTRL S.

Verifying the Current Communication Parameters

In TELCOM's Terminal Mode, you can check the current communication parameters by pressing **STAT** (F3) and ENTER whenever the `Telcom:` prompt is displayed. The Display will look similar to this:



```
M7I1E,10 pps
Telcom: Stat
M7I1E,10 pps
Telcom:
```

Changing Communication Parameters

If you need to change the Model 100 communication parameters to match the host system, follow this procedure:

1. Access TELCOM.
2. Press **STAT** (F3) (Stat). When you do this, the word `Stat` appears next to `Telcom` on the Display.
3. Type the new communication parameters in the following order:
    1. Baud Rate
    2. Word Length
    3. Parity
    4. Stop Bit
    5. Line Status
    6. Pulse Rate (Must be prefaced by a comma.)

Note that Pulse Rate, which is not a communication parameter but the speed at which the phone number is dialed, must have a comma before it when its changed. Also, you do not need to enter the Pulse Rate value if you wish to keep it at the current value even if you're changing the other parameters.

When changing communication parameters, you must type in each and every selectable parameter even if you don't want it changed. To leave a parameter at its current status (''unchanged value''), simply type in the current value as displayed on the Screen.

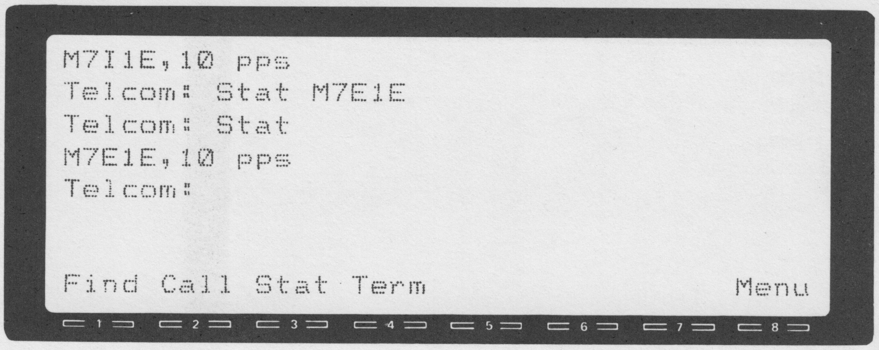
For instance, if the current status of TELCOM program is:

M7I1E,10

and you want to change the parity to Even:

1. Press **STAT** (F3).
2. Type: `M7E1E,10` ENTER

   The Telcom prompt will return to the Screen. Now, to check the new communication protocol press **STAT** (F3) and ENTER. The Display will show the new communication protocol.



```
M7I1E,10 pps
Telcom: Stat M7E1E
Telcom: Stat
M7E1E,10 pps
Telcom:


Find Call Stat Term                    Menu
```

Remember that you can use any of the allowable values listed in Table 11-2.

Entering Terminal Mode

Depending on the type of connection you are using, Terminal Mode may be entered two different ways:

- Automatically via auto-dialing
- Manually

The automatic method lets the Model 100 auto-dial and enter Terminal Mode simultaneously. This method works only when the Model 100 is connected directly to a modular telephone line.

When modular telephone lines aren't available, or when you're using an Acoustic Coupler, you must use the manual method to enter Terminal Mode.

Automatic Entry into Terminal Mode

To enter Terminal Mode while auto-dialing, you must first store the host system's phone number in the ADRS.DO file and follow it with the symbols < > — all of which should be enclosed within colons.

For instance, if CompuServe's phone number is 555-1234, store it as:

CIS :5551234 < >:

where **CIS** is *your* code for "CompuServe Information Service."

You may do the same with the telephone numbers of other host systems.

Then, proceed as follows:

1. Set the **ANS/ORIG** Switch (on the left side of the Computer) to **ORIG**.
2. Access TELCOM.
3. Press **FIND** (F1) and type the identifying label of the number you wish to find.

   In this case, press **FIND** and type `CIS` ENTER.
4. Press **CALL** (F2) when the phone number appears on the Display. (It is not necessary to lift the receiver when calling a host system.)

After the phone number has been auto-dialed, the Model 100 will produce a high-pitched tone to indicate that Terminal Mode has been entered.

In most cases, you will also hear the Model 100 "echo" what it hears. That is, if the phone you called is busy, you'll hear a busy signal. If the phone is ringing, you'll hear it ring.

At the same time, the last line on the Screen will display the "new" definitions of the Function Keys F1 through F8.

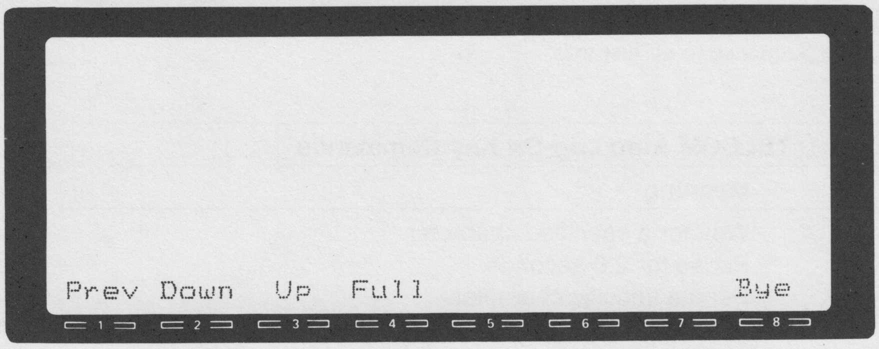


Figure 11-6. Terminal Mode Function Key Definition

Manual Entry into Terminal Mode

When you don't want to use the "Call" feature, follow these steps:

1. Access TELCOM.
2. Set the **ANS/ORIG** Switch (on the left side of the Computer) to the **ORIG** (for ORIGINATE).
3. Lift the receiver and dial the host system's phone number.
4. When the host answers, you will hear a high-pitched tone. Press **TERM** (F4).
5. If you're using an Acoustic Coupler, place the receiver in the cups; if not, hang up the receiver.

As soon as you press **TERM**, the Model 100 produces a high-pitched tone to indicate that it has entered Terminal Mode.

At that time, the last line on the Screen displays the "new" definitions of F1 through F8.

Once in Terminal Mode . . .

Whether you entered Terminal Mode manually or automatically, be sure to comply with the log-on sequence described in the information service user's guide. The user's guide is provided to you when you subscribe to a service.

Automatic Log-on

If you use an information service with some frequency, the log-on procedure may become time-consuming and tedious. Consequently, TELCOM gives you have the option of logging-on automatically.

The purpose of the log-on procedure is to provide some confirmation that you are authorized to access a host system. Most information services will provide you with a User ID and a Password to serve as confirmation.

The automatic log-on procedure should be used in conjunction with auto-dialing, enabling you to enter Terminal Mode and log-on all at once. For this reason, auto log-on, like auto-dialing, is available only when the Model 100 is directly connected to a modular telephone line using the ***Model 100 Modem Cable***.

The auto log-on procedure consists mainly of identifying the host computer's log-on prompts, and, sending the host the correct responses. The Key Commands listed in Table 11-5 are used as part of an Auto Log-on Sequence to do just that.

TELCOM Auto Log-On Key Commands

Key	Meaning
?	Wait for a specified character.
=	Pause for 2.0 seconds.
!	Send a specific character.
^	Causes the character after ^ to be sent as a "control" character (i.e., ^M is the same as pressing ENTER).

Table 11-5

Any character that is preceded by the ''caret'' symbol (^), is referred to as a Control Character. These characters are generated by typing SHIFT 6 and then the character itself. For instance SHIFT 6 C will send a ''control-C'' code.

To log-on automatically, you must include a combination of some of the Key Commands, along with your responses to the host's log-on prompts, within the greater and less than symbols (< >). This information must be created in TEXT and stored in the ADRS.DO file.

The information inside < > is referred to as the Auto Log-on Sequence.

You must also observe a sequential order in which the Key Commands and your responses will be stored. We will refer to this sequential order of events as the ''syntax'' of the Auto Log-on Sequence.

Auto Log-on Sequence Guidelines

In this section, we will describe the process for creating an Auto Log-on Sequence. However, before proceeding any further, you should note certain guidelines.

Syntax Follow a sequential order of events when creating the Auto Log-on Sequence. This means that questions (prompts) should be answered (responses) after they are asked.

In the case of CompuServe, for instance, you must first send a ^C then tell the Model 100 to wait for the first prompt (User ID:) before sending your response. Do this by using the Key Command ? (''wait for a specific character'') and a unique letter from the prompt (User ID:).

Selecting a Prompt Letter It is not necessary to include the entire phrase when telling the Model 100 to wait for a particular prompt.

That is, using the previous example, you don't need to say ? (''wait for'') User ID:. Instead, select any unique letter in the phrase User ID:. For instance, you could say:

? U ("wait for the letter U")
? s ("wait for the letter s")
? I ("wait for the letter I")

Observing Upper- and Lowercase Notice in the previous examples that whenever a letter was chosen from the log-on prompt to be included in the Log-on Sequence, it remained either upper- or lowercase — just as it was in the prompt.

Since the Model 100 distinguishes between upper- and lowercase characters in the Log-on Sequence, it is essential to use the correct form.

For instance, to ''tell'' the Model 100 to wait for the prompt User ID, you cannot use ?u when you really mean ?U.

Using the Key Command = (Pause for 2.0 Seconds) You should use a pause (=) only when the first action expected by the host is to receive a character from you.

Pause means ''pause for two seconds'' and is used to establish a good phone link with the host. As a general rule, it is wise to allow a certain amount of time before the log-on procedure begins (after the host system's telephone number has been dialed). By doing so, you can be certain that you have established contact with the host system before initiating telecommunications.

The Key Command **=** is intended for this purpose and should be the first Key Command in any Log-on Sequence where your first action is to send a Control Character to the host. This is the case with CompuServe since it requires that you send a ''control-C'' (^C) before logging-on.

Using the Key Command ! (Send a Specific Character) The Key Command **!** is used in a log-on sequence only when, as part of your responses, you are required to send the characters ? or =.

When the Key Command **!** precedes either **?** or **=**, they are no longer recognized as Key Commands but as a question mark and equals sign respectively.

For example, assume that your password is Billy?Boy. When creating your Log-on Sequence, the symbol **?** in your password must be distinguished from the Key Command **?** (wait for a specific character). This is done by inserting the Key Command **!** before ?. The Log-on Sequence would then be

?PBilly!?Boy

which means wait for P (for Password) from host and send response Billy?Boy. In other words, the Key Command **!** helps the Model 100 to distinguish between Key Commands such as **?** or **=** and responses which include the same symbols.

Creating an Auto Log-on Sequence

Although the procedure for creating a Log-on Sequence may seem lengthy and involved, in reality it is very simple. Also, once created, the Log-on Sequence lets you log-on to a host repeatedly and without effort.

In the following example, we'll use CompuServe to illustrate how a log-on sequence is created.

CompuServe's log-on prompts consist of **User ID:** and **Password:**. Both must be answered correctly before you are given clearance to the service. Assume, for the sake of this example, that your User ID is **98576,756**, and your Password is **Atom-Age**.

The Auto Log-on Sequence is stored (along with the host's name and telephone number) in the ADRS.DO file.

1. Begin by opening file ADRS.DO from the Menu and storing the host's name and telephone number. For instance:

   ```
   CIS :5551234<
   ```

   where **CIS** is your code for CompuServe Information Service and **555-1234** is your local CompuServe phone number.

   Note: If you haven't created file ADRS.DO see Chapter 10 for details.

2. Now, using the Key Commands, provide enough time for communications to be established effectively. Do this by using the Key Command = (''wait for 2.0 seconds'').

   Also, it is necessary to send a Control-C (^C) whenever attempting to access CompuServe. Control-C notifies CompuServe you are ready to initialize the log-on procedure. At this point, the Auto log-on sequence would look like this:

   ```
   CIS :5551234<=^C
   ```

3. After the ^C, you are ready to begin the log-on procedure.

   ''Tell'' the Model 100 to anticipate and wait for the first prompt, User ID. The Key Command ? (''wait for'') does this. Since it is not possible to include the entire phrase User ID after the Key Command ?, a single letter from this prompt must be chosen. Use **U**.

   The Auto log-on sequence would then be:

   ```
   CIS :5551234<=^C?U
   ```

4. Next, the rules for auto log-on dictate that you send the requested information — your User ID. Now, the Auto Log-on Sequence would appear as:

   ```
   CIS :5551234<=^C?U98576,756
   ```

5. Finally, it is necessary to ''ENTER'' this information so it can be acknowledged by the host computer. A Control-M (SHIFT 6 M) is used instead of the ENTER key.

   ```
   CIS :5551234<=^C?U98576,756^M
   ```

6. This process is repeated for the next prompt, **Password:**. Using the letter **P**, the Log-on Sequence for this would be:

   ```
   ?PAtom-Age^M>
   ```

   The complete Auto Log-on Sequence would then consist of:

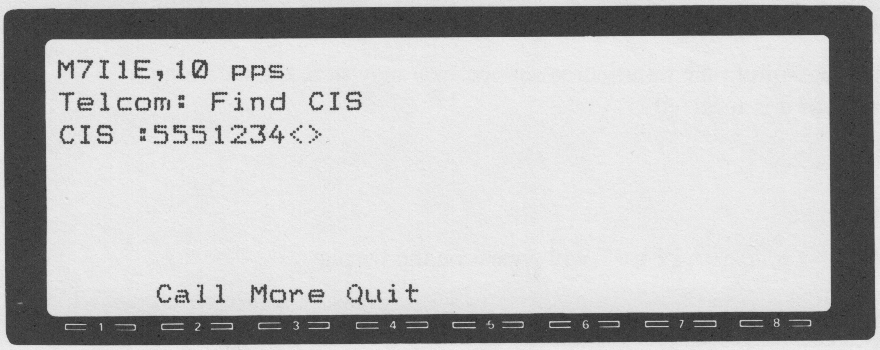
   ```
   CIS :5551234<=^C?U98576,756^M?PAtom-Age^M>:
   ```

Once you have created and stored an Auto Log-on Sequence in the ADRS.DO file, you can auto-dial, enter the Terminal Mode, and log-on to the host system from TELCOM all at once.

Using the above example:

1. Access the TELCOM Application Program.
2. Press **FIND** (F1).
3. Type `CIS` ENTER.

   The Display will show the host system's telephone number and the symbols < > without the Log-on Sequence. Don't be alarmed. This is simply a way to keep your ID number and password secret.

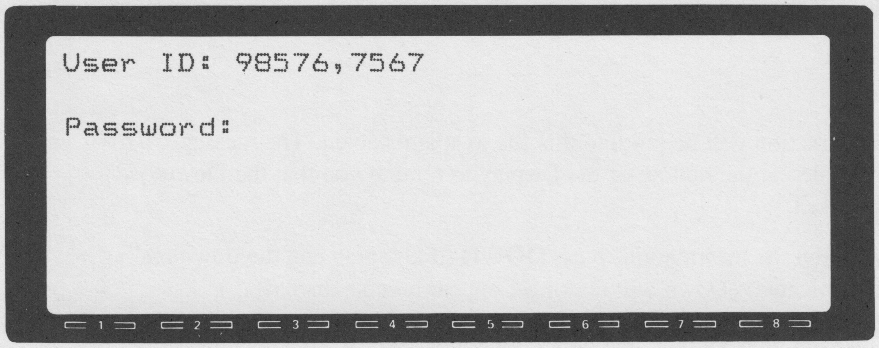


```
M7I1E,10 pps
Telcom: Find CIS
CIS :5551234<>




     Call More Quit
```

4. Press **CALL** (F2).

The telephone number digits will appear on the Screen as they are dialed. You do not need to pick up the telephone receiver. If the number is busy, a busy tone will emit from the Model 100; otherwise,you'll hear the "ring." After dialing is complete, the Model 100 will produce a high pitched-tone and the first prompt `User ID:` will appear. You will see your ID number appear next to the prompt.

Then the second prompt `Password:` will appear. However, you will not see Atom-Age after the prompt although the Model 100 sends your password to the host.



```
User ID: 98576,7567

Password:
```

You will then be logged-on to CompuServe.

Download and Upload

Two additional functions have been built into TELCOM — Download and Upload. "Download" implies that information is sent "down" to the Model 100 from another computer; "upload" implies that information is sent "up" to another computer from the Model 100.

When linked to an information service, you may not always have time to sit in front of the Model 100 and watch the Display. This is where the Download feature fits in. Download makes it possible to store incoming information in memory. This allows you to view or print the information later when you have time.

The Upload feature, on the other hand, allows you to transmit a previously created file to another computer. This might prove particularly useful when travelling.

For instance, instead of reciting a series of figures on the telephone to your office, simply type this information into a Text file, then "upload" it to your office system.

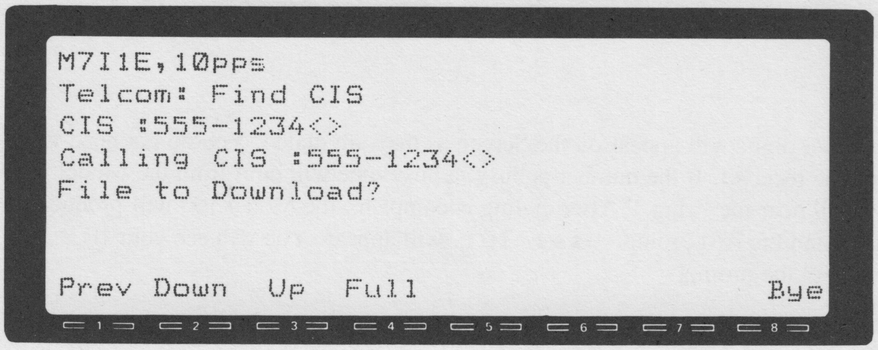
Using the Download Feature

After selecting a menu item from the information service, you may store the incoming information in memory as it is received.

To do this:

1. Press **DOWN** (F2).

   The message `File to Download?` will appear on the Display.



```
M7I1E,10pps
Telcom: Find CIS
CIS :555-1234<>
Calling CIS :555-1234<>
File to Download?


Prev Down   Up   Full                   Bye
```

2. Assign a file name (no longer than six characters) to the file which will store the incoming information.
3. Press ENTER.

   The incoming information will be put into this file as it is received. The message: `Down` will appear in reverse video at the bottom of the Display to remind you that the Download Function is being used.
4. When you have stored the information, press **DOWN** (F2) again and the downloading process will be terminated. (Down will no longer appear reverse displayed.)

If the Model 100 does not have enough memory available to store the file, the download will be automatically terminated. The file you created will contain as much information as was received. To download the entire file, return to the Main Menu, delete (''kill'') files you don't need, and repeat the download process.

To examine the received information:

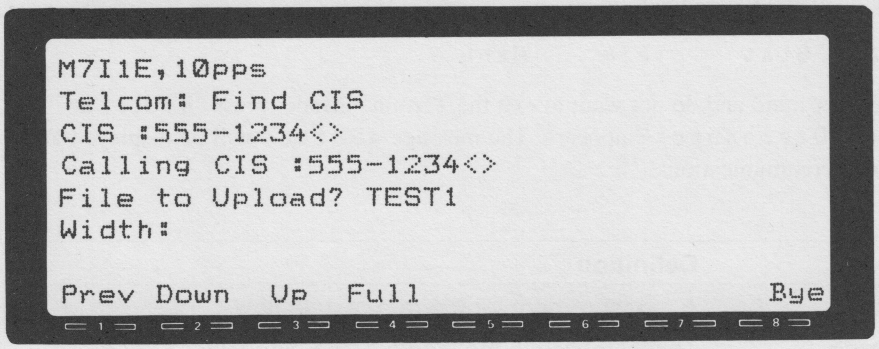
1. Log-off and exit Terminal Mode by pressing **BYE** (F8).
2. Press Y when the prompt `Disconnect?` appears. Press F8 to exit TELCOM and return to the Main Menu.
3. The Main Menu will then re-appear. Move the Cursor over the file name which you assigned to the file and press ENTER.

Once this file has been opened, you may use the Text Editing Functions to manipulate the information. You might find it necessary to do so since the file contents may include control characters that the host system uses when transmitting information. After editing, you can print the information if you wish.

Using the Upload Feature

You can send files you created with the TEXT program to a host system. To do so:

1. Enter TELCOM's Terminal Mode.
2. Press **UPLOAD** (F3). The message: `File to Upload?` will appear on the Screen.
3. Type the name of the file you wish to send and press ENTER.
4. The prompt: `Width?` will then appear. To send the file using the margin width of the Model 100 Display (40 characters), press ENTER.



```
M7I1E,10pps
Telcom: Find CIS
CIS :555-1234<>
Calling CIS :555-1234<>
File to Upload? TEST1
Width:

Prev Down  Up  Full                 Bye
```

If you choose to alter the width, simply type in the desired number for the width and press ENTER. The **Width** feature forces a carriage return to make your document come out just as wide as you want it. This makes ''word-wrap'' possible, preventing split words. Also, when UPLOADing to certain information services (CompuServe for instance) the host will ''choke'' when the input line exceeds 132 characters without a carriage return. **Width** ''imbeds'' a character return to prevent this.

Exiting Terminal Mode

Be sure to see your information service user's guide for the proper log-off procedure.

To terminate communications with an information service and return to the Main Menu:

1. Press **BYE** (F8).
2. The prompt `Disconnect?` will appear on the Display.
3. Press Y ENTER to disconnect.

   The status of the telecommunication parameters and the word Telcom: will appear. Also, the Function Key display will change to:

   ```
   Find Call Stat    Term    Menu
   ```

   If you change your mind and do not want to exit the Terminal Mode, press N (for "no") when the prompt `Disconnect?` appears. The message `aborted` will be displayed and you may resume communications.

Meaning	**Definition**
Baud Rate	A measurement for the rate of data flow.
Word Length	This establishes the length of the binary word to be transmitted per character.
Parity	A bit added to a group of bits which constitute a character as a method for error detection.
Stop Bit	The number of elements (bits) that mark the end of a character.
Line Status	This is a flow control (XON/XOFF) sequence that prevents information from flowing too rapidly for either the Model 100 or the host to assimilate.
Pulse Rate	Controls the rate at which a number is auto-dialed, either 10 or 20 pulses per second.

Table 11-5

PART III / MODEL 100 BASIC

This section of the manual will describe *in detail* Model 100 BASIC. If BASIC is new to you, be sure to read Chapters 12 through 15. These chapters will provide a general overview of BASIC including Data Types, Control Commands, and Input/Output information.

Once you're familiar with BASIC (or if you already are!), go on to Chapter 16 which is a description of how the Model 100 uses BASIC. Note that this chapter is organized in a "reference" format like a dictionary or any other reference manual. This means we've listed all the BASIC statements and functions in alphabetical order and described a "keyword's" syntax and allowable values. We've also provided a brief description and short examples. (For more complex examples of BASIC programs, see the Appendices.)

This section isn't meant to teach you how to program in BASIC. If you want to learn more about programming, use the specific information provided in Chapter 16, then refer to ***Getting Started With TRS-80® BASIC*** (26-2107).

12 / Model 100 BASIC Overview

Model 100 BASIC is an easy to use, extended version of the BASIC programming language. The word ''BASIC'' stands for ''Beginners' All-purpose Symbolic Instruction Code.'' It has been specially designed to take advantage of all of your Model 100's advanced features.

Model 100 BASIC does not produce a low-level, machine language translation but instead executes your programs directly. In technical terms, it is an ''interpreter'' rather than a ''compiler.'' This makes Model 100 BASIC especially powerful for interactive use during program development and debugging.

Model 100 BASIC offers all of the standard features of the BASIC language, plus several important additions:

- A wide range of Input/Output statements (I/O), which let you access the I/O devices (the LCD screen, the RS-232C port, the printer, and so on).
- Commands to generate ''music'' from the Model 100's five octave sound generator.
- Special commands which allow your BASIC programs to call machine language subroutines.
- Special ''interrupt'' instructions which let your program handle special conditions originating from error conditions, clock conditions, and communication line events.

Starting Up BASIC

To start up Model 100 BASIC from the main menu, use the arrow keys to position the Cursor on top of the word `BASIC` and press (ENTER).

Alternatively, if you want to run a BASIC program which is stored in RAM, simply position the Cursor on top of the program name and press (ENTER). Your Model 100 ''recognizes'' BASIC programs and will start up BASIC and run the BASIC program you selected.

Modes of Operation

Model 100 BASIC has three modes of operation:

- **Command Mode** for typing in program lines and immediate lines
- **Execute Mode** for execution of programs and immediate lines
- **Edit Mode** for editing program lines

Command Mode

Whenever you enter the Command Mode (either from the Menu or after a program ends), BASIC displays the word **Ok**, followed on the next line by a blinking Cursor at the beginning of the current *logical* line. A logical line is a string of up to 255 characters terminated by a carriage return (stored when you press (ENTER)). A *physical* line, on the other hand, is one line on the Display. It contains 40 characters.

In the Command Mode, BASIC does not process your input until you press (ENTER). While typing in the line, you may use (BKSP) or (←) to delete characters. You may also completely cancel the line by pressing (BREAK).

Interpretation of an Input Line BASIC always ignores leading spaces in a line — it jumps ahead to the first non-space character. If this character is *not* a digit, BASIC treats the line as an *immediate* line. If it is a digit, then BASIC treats the line as a *program* line.

You may use either upper- or lowercase for entering commands — BASIC converts all lowercase (except what is in quotes) to uppercase.

Immediate Line An immediate line consists of one or more BASIC commands, separated by colons. The line is executed as soon as you press ENTER. For example,

```
Ok
CLS: PRINT "The square root of 2 is "; SQR(2)
```

is an immediate line. When you press ENTER, BASIC clears the screen (CLS), then PRINT's the message **The square root of 2 is**, and finally prints the square root (SQR) of 2.

Program Line A program line consists of a line number in the range 0 - 65529, followed by one or more statements separated by colons. When you press ENTER, BASIC stores the line in memory, along with any other program lines which you have typed in. The Computer doesn't execute the lines until you type RUN. For example,

```
Ok
100 CLS: PRINT "The square root of 2 is "; SQR(2)
```

is a program line. When you press ENTER, BASIC stores it in memory. To execute this line (and any other lines stored in memory), type:

```
RUN ENTER
```

Special Keys and Instructions in the Command Mode Since Model 100 BASIC is an "interpreter," any immediate line can also be a program line, and vice versa. However, there are certain commands which are particularly useful as immediate lines. See Table 12-1.

COMMAND	OPERATION
CLEAR	Clears all variable values
CLS	Clears the screen
CONT	Continue execution after a STOP command or BREAK
FILES	Prints the data and program files stored in RAM
FRE	Returns amount of memory available to BASIC
HIMEM	Returns address of highest memory available to BASIC
IPL	Defines a BASIC program to run upon power up
KILL	Erases a RAM file
LCOPY	Copies the screen text to the printer
LIST	Lists the current program onto the screen
LLIST	Lists the current program onto the printer
LOAD	Loads a BASIC program
MAXRAM	Returns address of highest available memory
MENU	Returns to the Model 100 menu
MERGE	Combines two BASIC programs
NAME	Renames a RAM file
NEW	Erases the current program
POWER	Controls the automatic power off feature
?	Abbreviation for PRINT
SAVE	Saves the current program (for instance, to RAM)

Table 12-1

Each of these instructions is discussed in detail under **"BASIC Keywords."**

In addition, there are several keys that when pressed perform an operation. See Table 12-2.

KEY	OPERATION
LABEL	Prints the definitions of the function keys
PRINT	The equivalent of typing in "LCOPY"
SHIFT PRINT	The equivalent of typing in "LLIST"
F1	Types Files ENTER
F2	Types Load "
F3	Types Save "
F4	Types Run ENTER
F5	Types List ENTER
F8	Types Menu ENTER

Table 12-2

Note that you can define any of the Function Keys for any purpose. See KEY under **"BASIC Keywords."**

Execute Mode

Whenever BASIC executes statements, either as immediate lines or as program lines, it is in the *execute* mode. Unless the program requires your input, the Computer is under the control of BASIC.

Special Keys in the Execute Mode

While BASIC is executing a program or command line, pressing certain keys cause an operation. See Table 12-3.

KEY	OPERATION
BREAK	Stops execution of the current command. You can restart many commands where they left off by typing CONT ENTER.
PAUSE	Temporarily stops execution of the current command. To continue, simply press PAUSE again. This is particularly helpful when the screen is changing rapidly, for example, on a LIST.

Table 12-3

You can also define any of the eight function keys as special "interrupters." See **ON KEY** under **"BASIC Keywords"** for details.

Edit Mode

While you are in the Command Mode, you have access to the Model 100 Text Editor.

To enter the Edit Mode, type **EDIT** followed by the range of line numbers you wish to edit. The line number ranges are shown in Table 12-4.

null	Edits the entire program. For example, **EDIT**
line1-line2	edits from *line1* to *line2,* inclusive. For example, **EDIT 100-200**
-line2	edit from beginning to *line2*. For example, **EDIT -100**
line1-	edit from *line1* to the end of the program. For example, **EDIT 100-**
.	Edits last accessed line number (last edited, entered, listed, and so on). For example, **EDIT .**

Table 12-4

To exit the Text Editor, press (F8). Note that the text you edit must consist of valid BASIC lines — valid line numbers for each line and no lines greater than 255 characters.

For details of how to use the Text Editor, see Part II of this manual.

Note: When entering and leaving the Text Editor, BASIC must convert the program lines, which are stored in a "tokenized" format, into an ASCII format. Depending on the size of the program, this may take up to two minutes. You can minimize this wait time by only editing short amounts of text.

A Sample BASIC Program

A BASIC program consists of one or more logical lines, each line beginning with a line number followed by one or more BASIC commands. BASIC allows line numbers from 0 to 65529 inclusive. The program lines can include up to 255 total characters including the line number, and may be broken down into two or more physical lines.

For example, here is a short BASIC program:

```
100 CLS: PRINT "Here's a short BASIC program!"
110 SUM = 0
120 FOR I=1 TO 100
130 SUM = SUM + I
140 NEXT I
150 PRINT "Sum of 0-100 = ";SUM
```

Normally, when BASIC executes a program, it handles the *commands* one line at a time, starting at the first and proceeding to the last. (Certain commands, however, allow you to change this sequence. See "**Control Commands**.")

A *command* is a complete instruction to BASIC. It tells the Computer to perform some operation. If the operation involves data, then the command may include that data. For example,

```
PRINT "The square root of two is ";SQR(2)
```

is a complete command. The number 2 is data, as is the string inside the quote marks. The operations are:

- Displaying the message inside the quote marks
- Computing the square root of 2
- Displaying the resultant value

Most commands are made up of *keywords*. These *keywords* have special meaning to BASIC, hence you must be sure not to use any of these words as variable names.

13 / Data Types and Data Manipulation

Model 100 BASIC can handle two kinds of data:

- **Numbers**, representing quantities, and subject to standard mathematical operations.
- **Strings**, representing sequences of characters, and subject to special non-mathematical string operations.

Numeric Data

BASIC allows three types of numbers — double-precision, single-precision, and integers. You can declare the type of a number, or let BASIC assign a type. Each type serves a specific purpose in terms of precision, speed, and arithmetic operations, and range of possible values. By default, Model 100 BASIC considers all numbers as double-precision.

Double-Precision Numbers

Double-precision numbers consist of up to 14 significant digits, plus a decimal point. A double precision number requires eight bytes of memory for storage, and may be represented with exponential notation, with exponents ranging from −64 to 62. This gives them an effective range of $\pm 1 \times 10^{62}$ and $\pm 1 \times 10^{-64}$. For example,

1.3402100054 **3.1415926535898** **1.44343455331D−40**

can all be stored as double-precision numbers. Note that the letter **D** represents a double-precision number in exponential notation. For example, 1.443455331D−40 means $1.443455331 \times 10^{-40}$.

Single-Precision Numbers

Single precision numbers consist of up to six significant digits, plus a decimal point. A single precision number requires four bytes of memory for storage, and may be represented with exponential notation, with exponents ranging from −64 to 62. This gives them an effective range of $\pm 1 \times 10^{62}$ and $\pm 1 \times 10^{-64}$. For example,

100.003 **−23.4212** **1.4432E6** **4.552E−14**

can all be stored as single precision numbers. Note that the letter **E** represents a single precision number in exponential notation. For example, 1.4432E6 means 1.4432×10^{6}.

Integer Numbers

Integers are the most efficient type of number in terms of calculations and storage. They lie in the range of −32768 to 32767, and require two bytes of memory for storage. Negative numbers are stored in two's complement form. Note that you may not use decimal points in integers. For example,

1 **32000** **−2** **500** **−12345**

can all be stored as integers.

String Data

String data consists of a sequence of characters. These characters may be numbers, letters, and special characters like #, $ and %. Strings are stored by the ASCII code of the characters in the string, with each character using one byte of memory for storage. For example,

5641 Country Lane

can be stored as a string of 17 characters.

Strings may vary in length from 0 to 255 characters. Strings with zero length are called *null* or *empty* strings.

Representing Data

BASIC recognizes data in two forms — either directly, as *constants*, or by reference, as *variables*.

Constants

Constants are values which do not change during the course of the execution of the program. When BASIC encounters a data constant in a statement, it must determine the type of the constant — either string or numeric:

- If a data constant is enclosed in double-quotes, BASIC considers it to be a string. For example:

  "ABC" **"Enter Check Number"** **"139.4"**

- If the constant is not in quotes, BASIC considers it to be a double-precision number. For example,

  1234 **1.234005993334** **7.567E+10**

Variables

A variable represents a storage place in memory. Unlike a constant, a variable's value can change throughout the course of a program. By default, all variables are double precision type. You may change this with type declaration commands and tags, as described later in this chapter.

Variable names consist of a letter, followed by one or more numbers or letters, although only the first two characters of the variable name are significant. For example,

A **B3** **ZZ** **AS**

are all valid, unique variable names. However,

SU **SUPER** **SUPERSCRIPT**

represent the same variable, namely, SU.

Additionally, certain combinations of letters are reserved by BASIC as ''keywords'' or ''reserved words,'' which have special meaning to the BASIC Interpreter. You may not use such combinations as variable names. For example,

TOTAL **LAND** **NAME** **LENGTH**

cannot be used as variable names because they contain the keywords TO, AND, NAME, and LEN, respectively.

Arrays

All of the variables mentioned above are simple variables. They can only refer to one data item.

Variables may also be dimensioned and subscripted so that an entire list of data can be stored under one variable name. Such structures are called arrays. You may access each element of the array as if it were a single variable name. For example, you may use each of these elements to store a separate data item, such as:

```
ACCNUM(0) = 1223
ACCNUM(1) = 1224
ACCNUM(2) = 1225
ACCNUM(3) = 1226
```

In this example, ACCNUM is a *one-dimensional* array, since each element contains only one subscript. An array may also be two-dimensional, with each element containing two subscripts. You can think of a two dimensional array as a *table*, with one subscript referring to the row, and the other referring to the column.

For single dimensional arrays larger than 11 elements, and for all multi-dimensional arrays, BASIC requires that you declare the dimensions of arrays before you use them, through the **DIM** (dimension) statement. It has the following syntax:

DIM *array name1 (dimension list1), . . .*

where dimension list is the size of the array in each dimension. For example,

```
DIM MN$(12)
```

defines MN$ to be a string array of **13** elements, MN$(0) through MN$(12). You may include more than one array per DIM statement, for example,

```
DIM MN$(12), JMPTB(10,100)
```

defines MN$ to be a string array of 13 elements, MN$(0) through MN$(12), and JMPTB to be an array of 1111 elements, JMPTB(0,0) through JMPTB(10,100).

The number of dimensions allowed by Model 100 BASIC is dependent solely on the amount of memory you have remaining. If, for example, you have 20000 bytes of memory free, you may define an integer array as:

```
DIM ARR%(5,5,5,5,5)
```

without error, since integers each take up two bytes of memory, so this array contains 15552 elements (6^5 X 2). However, if you try to dimension a double precision array with the statement:

```
DIM DBA(5,5,5,5,5)
```

you'll generate an error since this array requires 62208 bytes of memory. (Remember that double precision numbers require eight bytes of memory each!)

A note about subscripts: Subscripts may be constants or numeric expression. However, they may not be negative or greater than the dimensioned size or else a BS error (Bad Subscript) occurs.

Type Declaration Tags

When BASIC encounters a variable name in the program, it may classify it as either string, integer, single-precision, or double-precision. Initially, BASIC classifies all variable names as double-precision.

You may assign different attributes to variables based on the first letter of the variable. You do this with definition statements at the beginning of your program. DEFINT defines variables as integer, DEFSNG defines variables as single-precision, DEFDBL defines variables as double-precision, and DEFSTR defines variables as string.

For example:

```
DEFSTR L, A-C
```

defines all variables which start with L, A, B, and C as string variables. After this statements, the variables L, LS, LG, AA, AB, BD, C1 can only hold string values.

You may also use type declaration tags as a suffix to variable names. Type declarator tags take precedence over DEF commands. These tags are Listed in Table 13-1.

TAG	OPERATION
%	Declares the variable to be integer. For example, `A%` `INC%` `NUM%` `Z1%` are all integer variables.
!	Declares the variable to be single-precision. For example, `PER!` `AVE!` `T1!` `CNT!` are all single-precision variables
#	Declares the variable to be double-precision. For example, `PER#` `AVE#` `T1#` `PI#` are all double-precision variables.
$	Declares the variable to be string type. For example, `NM$` `ADR$` `MN$` `ST$` are all string variables.

Table 13-1

Expressions

Expressions are composed of *operands* and *operators*. Operands are simply the constants and variables of your program, while operators are special instructions which tell the Computer to perform a certain operation on the operands.

You use expressions as part of *assignment statements* (discussed later this chapter), in *relationship tests* (also discussed later this chapter), or in *Input/Output statements*.

The simplest expression consist of only one term, either a constant or a variable. For example,

A **14.44** **B$** **ACT!**

You may also use more complex expressions, consisting of several operands and operators. For example,

A + C **1.223/(33+IVAL)** **B$ + " " + MN$** **A < B**

We can divide expressions into four broad categories — *numeric expressions, string expressions, relational expressions,* and *logical expressions*. *Numeric expressions* are composed entirely of numeric constants, variables, functions, and numeric operators. *String expressions* are composed entirely of string constants, variables, functions, and the string operator.

Relational expressions test the relationships between numeric or string expressions, and hence can be composed of relational operators and either numeric expressions or string expressions. The computer evaluates relational expressions as either "true" or "false". (Numerically speaking, relational expressions return the values 0 for false and −1 for true.)

Logical expressions deal with true/false conditions and Boolean operations, and are composed of logical operators, relational expressions, and numeric expressions. The result of logical expressions is numeric.

Numeric Expressions

Numeric expressions are composed of numeric operators, operands, and functions. Numeric operators can be either *binary* or *unary*. Binary operators specify an action on two operands, while unary operators specify an action on one operand.

The binary numeric operators are shown in Table 13-2.

OPERATOR	MEANING
+	Addition
−	Subtraction
*	Multiplication
/	Division
\	Integer Division
^	Exponentiation
MOD	Modulus Arithmetic

Table 13-2

(Note: To enter \, press GRPH and − simultaneously. To enter ^, press SHIFT and 6 simultaneously.)

The unary numeric operators are:

+	Positive sign
−	Negative sign

Binary numeric expressions have the form:

operand1 operator operand2

while unary numeric expressions have the form:

operator operand

You can mix numeric data types within an expression. BASIC automatically converts values to double precision. (**Note:** BASIC does **not** convert the actual value stored in memory, but rather uses a "working" value, stored in memory only until the operation is complete.)

Examples

Addition:

25 + 15.3334

results in the value 40.333400000000.

Subtraction:

20.443 − 4

results in the value 16.443000000000.

Multiplication:

300 * 100

results in the value 30000.000000000.

Division:

10 / 3

results in the value 3.3333333333333.

Integer Division:

10 \ 3

results in the value 3.0000000000000 (the decimal portion of the quotient is dropped).

Modulus Arithmetic:

10 MOD 3

results in the value 1.0000000000000 (the "remainder" of the 10 \ 3).

Exponentiation:

10 ^ 3

results in the value 1000.0000000000.

Unary Plus:

+ 13.554

results in the value 13.554000000000.

Unary Minus:

−4.54454

results in the value −4.5445400000000.

Numeric Functions Numeric functions are special BASIC operators which take operands, perform some operation on them, and return a numeric result. Usually, you must specify the operands (often called the "arguments") parenthetically after the function name. Some functions require no arguments.

For example, SQR is a numeric function returning the square root of its argument. The expression **SQR(25)** returns a value of 5.

The BASIC numeric functions are shown in Table 13-3.

FUNCTION	OPERATION
ABS	Return absolute value
ASC	Get ASCII code
ATN	Compute arctangent
CDBL	Convert to double-precision
CINT	Convert to integer
COS	Compute cosine
CRSLIN	Return line position of cursor
CSNG	Convert to single-precision
EOF	Returns end-of-file status
ERL	Returns the line number of the last error
ERR	Return the error code of the last error
EXP	Compute natural exponential
FIX	Truncate to whole number
FRE	Return current amount of available memory
HIMEM	Return highest address of memory available to BASIC
INP	Return a value from a CPU port
INSTR	Search a string for a substring
INT	Convert to integer
LEN	Compute length of string
LOG	Compute natural logarithm
LPOS	Return column position of print head
MAXRAM	Return the size of your Model 100's memory
PEEK	Return value at a memory address
POS	Return column position of cursor
RND	Return pseudo-random number
SGN	Return algebraic sign
SIN	Compute trigonometric sine
SQR	Compute square root
TAB	Position cursor or print head on a print command
TAN	Compute tangent
VAL	Convert string to numeric
VARPTR	Return memory address of a variable

Table 13-3

See **"BASIC Keywords,"** for specific details of each numeric function.

String Expressions

Strings expressions are composed of the *string operator, string operands,* and *string functions.* BASIC has only one string operator, the "+" symbol. This operator tells the computer to "concatenate" *operand2* onto *operand1*. (Concatenation means to "connect in a series".) For example,

A$ + B$

results in the characters of B$ connected onto the end of A$. Note that unlike numeric addition, the order of the operands **is** significant. For example,

"Wed" + "nesday"

and

"nesday" + "Wed"

result in two different strings — "Wednesday" and "nesdayWed."

Since BASIC allows strings of up to 255 characters in length, you will get an error if the resulting string is longer than 255 characters.

String Functions String functions take operands, perform some operation on them, and return a string result. You specify the operands (often called the *arguments*) parenthetically after the function name. Some functions don't require arguments — they simply return some string value.

For example, CHR$ converts its numeric argument to the ASCII character represented by the argument. The function CHR$(65) returns an "A".

The BASIC string functions are shown in Table 13-4.

FUNCTION	OPERATION
CHR$	Returns ASCII character
INKEY$	Returns any keyboard key currently pressed
INPUT$	Returns a number of characters from the keyboard
LEFT$	Returns left portion of a string
MID$	Returns middle portion of a string
RIGHT$	Returns right portion of a string
SPACE$	Returns a string of spaces
STR$	Converts numeric to string type
STRING$	Returns a string of characters

Table 13-4

See **"BASIC Keywords,"** for specific details of the string functions.

Relational Expressions

Relational expressions compare two numerical or string expressions. If the comparison is true, then the computer evaluates the expression as -1. If the comparison is false, then the computer evaluates the expression as 0.

BASIC uses the following relational operators shown in Table 13-5.

OPERATOR	FUNCTION
<	Less than
>	Greater than
=	Equal to
<> or ><	Not equal to
=< or <=	Less than or equal to
=> or >=	Greater than or equal to

Table 13-5

For comparing numeric expressions, the BASIC compares the values of each numeric expression. Hence $5.4 < 7.9999$ is evaluated as *true* and $5.4 = 4.5$ is evaluated as *false*.

For comparing string expressions, the BASIC compares the ASCII code of each character of the string, from left to right. For example, "A" < "C" is true, since 65, the code for "A", is less than 67, the code for "C". Further examples are:

`"abc" > "ABC"` is true
`"ABCD" > "ABC"` is true
`"A " = " A"` is false

Usually, you will use relational expressions as part of an **IF . . THEN . . ELSE** statement. An **IF . . THEN . . ELSE** statement tests a condition, and based on the truth or falsehood of the statement tells BASIC to take a particular action. For example,

```
IF A < B THEN GOTO 100 ELSE GOTO 200
```

If A is less than B, the relationship is true, and the program proceeds to statement 100. If A is equal to or greater than B, the relationship is false, and the program proceeds to statement 200. **IF . . THEN . . ELSE** statements are discussed in further detail under **"BASIC Keywords."**

Logical Expressions

Logical expressions perform Boolean logic operations on or between operands. Boolean logic deals with *bit by bit* comparisons between numbers. You can use logical expressions to combine relational expressions.

There are six logical operators, five of which are binary and one which is unary. Table 13-6 summarizes the action of the logical operators:

Operator	Meaning of Operation	First Bit	Second Bit	Result Value
AND	When both bits are 1, the result is 1	1 1 0 0	1 0 1 0	1 0 0 0
OR	When either bit is 1, the result is 1	1 1 0 0	1 0 1 0	1 1 1 0
XOR	When one bit is 1 and the other is 0, the result is 1	1 1 0 0	1 0 1 0	0 1 1 0
EQV	When both bits are 1 or both bits are 0, the result is 1	1 1 0 0	1 0 1 0	1 0 0 1
IMP	Result is 1 unless the first bit is 1 and the second bit is 0	1 1 0 0	1 0 1 0	1 0 1 1
NOT	Result is 1 if bit is 0	1 0	(unary)	0 1

Table 13-6

The computer converts the operands to integers before performing the logical operation. Note that when all the bits of a binary number are 1's, then the decimal value is -1, or "true," and when all the bits of a binary number are 0s, then the decimal value is 0, or "false." (Remember that integers are 16 bit two's complement numbers!)

Because of these relationships, you can perform logical operations between relational expressions, and evaluate the combined relation. For example,

1 < 7 OR 4 > 10.2

1 is less than 7 but 4 is not greater than 10.2. However, since the logical operator is OR, the expression has the value:

true OR false

or, in binary terms,

1111 1111 1111 1111 OR 0000 0000 0000 0000

The result is ''true'' (that is, 1111 1111 1111 1111 in binary or −1 in decimal).

Normally, you will use logical expressions as part of an IF . . . THEN . . . ELSE command. For example

```
IF A<C AND B$ <> "Sort" THEN GOSUB 1000
```

Assignment Statements

BASIC provides a means for ''assigning'' a value to given variable via the ''assignment'' statement. An assignment statement takes the form:

variable name = expression

Assignment statements tell the Computer to replace the previous value of variable name with the value of expression. For example,

A = A + 1

is a valid BASIC statement. It tells the computer to add 1 to the value currently in A, and replace this current A with the resulting sum.

More Complex Expressions

BASIC allows you to combine several short expressions into one larger expression. For example,

```
C = A + B
C = C / 5
C = C + 3
```

is equivalent to:

C = (A + B) / 5 + 3

When evaluating compound expressions, BASIC performs the operations according to a specific hierarchy, so that the results are always predictable. Table 13-7 summarizes this hierarchy, from highest to lowest:

Parentheses
^
+, − (unary plus and minus)
*,/
MOD
+, −
<,>,=,=>,<=,><
NOT
AND
OR
XOR
EQV
IMP

Table 13-7

Within an expression, operators on the same level are evaluated from left to right, with the exception of parentheses, which are evaluated from inside to outside.

Data Conversion

Since the Model 100 BASIC lets you use several different types of data within expressions and assignment statements (single-precision, string, and so on), you need to be aware of certain ''conversion'' rules which determine type of the value resulting from an expression.

BASIC lets you convert any numeric data type to any other numeric data type (with some restrictions) simply through assignment statements. However, you may *not* convert numeric type data to string type data, or vice versa. (BASIC does provide two functions, STR$ and VAL, which will convert numbers to their string equivalent, and vice versa, as well as other functions which convert one numeric type to another.)

Single- or Double-Precision to Integer

BASIC returns the integer value left after truncation of the numbers to the right of the decimal point. Since integers can only hold values greater than or equal to −32768 and less than +32768, the single or double precision number must be in this range.

Examples

A% = −10.5

assigns A% the value −10.

A% = 32767.9

assigns A% the value 32767.

A% = 2.511195534432

assigns A% the value 2.

A% = −32768.6

produces an Overflow Error (−32769 is out of range for an integer value).

Integer to Single- or Double-Precision

BASIC returns the integer value with an appropriate number of zeroes added to the right of the decimal point.

Examples

A! = 32767

Assigns A! (a single-precision variable) the value 32767.0

A# = −1234

Assigns A# (a double-precision variable) the value −1234.0000000000

Double-Precision to Single-Precision

BASIC rounds the value of the double-precision number to fit the single-precision value, using "4/5" rounding (round values from 1-4 down, round values from 5-9 up).

Examples

A! = 1.2345678901234567

Assigns A! the value 1.23457.

A! = −1.24854499999

Assigns A! the value −1.24854

Single- to Double-Precision

BASIC adds sufficient trailing zeroes to the right of the decimal point to create a double-precision number.

Examples

A# = 1.5

Assigns A# the value 1.500000000000.

14 / Control Commands

Unless told otherwise, the Computer executes your BASIC program one line at a time, starting from the lowest numbered line and continuing up to the highest numbered line. However, many times you may want to execute the same instructions several times; other times, you may want to skip some instructions. BASIC, like all other programming languages, handles such program branching with *control commands*.

Simple Control Commands

BASIC uses four commands to cause branching in a program. These commands are GOTO, GOSUB, FOR . . . NEXT, and CALL.

GOTO

GOTO has the form:

GOTO *line number*

When BASIC executes a GOTO, it branches to line number just as if line number physically followed the GOTO command. For example,

```
10 INPUT "Enter another number:";N
20 PRINT "You entered a:";N
30 GOTO 10
```

Every time that BASIC gets to line 30, it executes the GOTO and jumps back to line 10 and starts over.

Often times, you'll use the GOTO command as part of a ''conditional'' command. BASIC conditionals are IF . . . THEN . . . ELSE commands and ON expression commands. For example,

```
10 INPUT "Answer yes or no (Y/N)";AN$
20 IF AN$ = "Y" THEN GOTO 100
30
.
.
.
```

If you answer line 10's input prompt with **Y**, then BASIC jumps to line 100; otherwise, BASIC continues with line 30.

The ON expression statement selects a line number to GOTO from a list of line numbers, based on the value of an expression. For example,

```
10 INPUT "1, 2, or 3";AN
20 ON AN GOTO 100,200,300
30
```

If you respond to the input prompt of line 10 with a 1, then line 20 tells BASIC to jump to line 100, a 2 tells BASIC to jump to line 200, and a 3 tells BASIC to jump to line 300. If you respond with a number other than 1,2 or 3, then BASIC continues with line 30.

For more information on IF . . . THEN . . . ELSE and ON expression, see **"BASIC Keywords."**

GOSUB

Like GOTO, GOSUB has the form:

GOSUB *line number*

However, a GOSUB command is actually a *call* to a *subroutine*. When BASIC jumps to the subroutine, after executing a number of lines, it expects to find a RETURN command. When BASIC executes the RETURN command, it jumps back to the command immediately following the GOSUB command. For example,

```
10 GOSUB 100: PRINT "Average = ";AVE
.
.
100 SUM = A + B + C + D
110 AVE = SUM / 4
120 RETURN
```

Line 10 calls the subroutine beginning in line 100. The subroutine adds up four values and finds the average. Line 120 contains the RETURN command. When BASIC executes this line, it jumps back to middle of line 10 (to the PRINT command) and begins execution there.

Like GOTO, you may often use GOSUB as part of a IF . . . THEN . . . ELSE command or a ON expression command. See **"BASIC Keywords"** for a description of these commands.

FOR . . . NEXT

The FOR . . . NEXT structure causes a repetition of number of BASIC lines, one or more times. It has the form:

FOR *variable = initial value* **TO** *final value* **STEP** *increment*
.
.
NEXT *variable*

The first time BASIC executes the FOR command, it sets *variable* equal to the *initial value*. It then executes the commands, all the way to the NEXT command. There, BASIC adds *increment* to *variable*, then compares *variable* with final value. If *variable* has reached the *final value*, BASIC simply continues with the next line. However, if *variable* has not yet reached final value, BASIC branches back to the line following the FOR command.

FOR commands are very convenient for handling arrays. For example,

```
10 SUM = 0
20 FOR I=1 TO 10
30 SUM = SUM + A(I)
40 NEXT I
```

This FOR . . . NEXT set sums elements one through ten of the array A.

You may "nest" one FOR . . . NEXT set inside of another FOR . . . NEXT set. This is helpful in dealing with multiple dimensioned arrays. For example,

```
10 SUM = 0
20 FOR I=1 TO 10
30 FOR J=1 TO 20
40 SUM = SUM + A(I,J)
50 NEXT J
60 NEXT I
```

This routine sums the elements of an array A, from A(1,1) to A(10,20). Note that for every iteration of the outer loop (FOR I= . . .), the inner loop (FOR J= . . .) is executed 20 times.

Note that increment may be negative and final value may be less than initial value. In such a case, the loop simply counts "backwards."

CALL

CALL is another form of a subroutine call, except that instead of forcing a jump to a BASIC subroutine, it forces a jump to a *machine-language* subroutine. CALL has the form:

CALL *address, accumulator value,* **HL** *value*

where *address* is the memory address of the program, and the *accumulator value* and HL *value* are *parameters*. Since machine level programs can't access variables by name like BASIC does, you must "pass" any values that the subroutine may need via the accumulator and HL registers. *accumulator value* may range from 0 to 255, and HL *value* may range from −32768 to 65535.

A note about machine-language subroutines: Machine-language subroutines are made up of special sequences of numbers which tell the Computer what operations to perform. When BASIC interprets your programs, it converts the BASIC commands to many simpler machine-language commands.

Using machine-language subroutines has advantages as well as disadvantages. These routines run much faster than a BASIC subroutine, since they don't require any interpretation by BASIC. Also, some applications require operations which simply aren't available with BASIC. The major disadvantage of machine level subroutines is that they require a knowledge of 80C85 machine-language codes.

Interrupt Commands

The control commands discussed in the first part of this chapter are immediately executed whenever BASIC encounters them in your program. However, in some instances, it may be convenient to force a call to a subroutine while BASIC is executing unrelated code.

For example, you may want BASIC to perform some operations at ten o'clock. But rather than tie up the Computer in a loop which constantly checks the time, you want the Computer to work on some other job.

By defining an *interrupt* condition (in this case a particular time), when the condition is reached, your program can stop what its doing and branch to some subroutine. When the subroutine is completed, BASIC returns to the point in the program where it was interrupted.

There are five interrupt definition commands (see Table 14-1).

Command	Operation
ON COM	Calls a subroutine when the Computer receives data over the RS-232C line.
ON ERROR	Branches to an error handling routine if some error occurs while executing the program.
ON KEY	Calls a subroutine if you press one of the eight Function Keys.
ON MDM	Calls a subroutine when the Computer receives data over the modem.
ON TIME$	Calls a subroutine when the clock reaches a certain time.

Table 14-1

Within your program, you may *enable* or *disable* the interrupt function. For BASIC to register an interrupt, you must first issue an ON command. Likewise, you may tell BASIC to ignore any interrupt with an OFF command. Finally, you may tell BASIC to ignore an interrupt, but "remember" that the condition occurred, with the STOP command. Enable/disable commands are listed in Table 14-2.

Enable	Disable	Disable/Remember
COM ON	COM OFF	COM STOP
MDM ON	MDM OFF	MDM STOP
KEY ON	KEY OFF	KEY STOP
TIME$ ON	TIME$ OFF	TIME$ STOP

Table 14-2

(ON ERROR requires no enable or disable.)

With the exception of ON ERROR, all of the interrupt commands call subroutines via GOSUB statements, and end with a RETURN command. ON ERROR uses a simple GOTO for branching, and the error handling routine ends with a RESUME command.

Here is a short example of an interrupt using ON TIME$.

```
10 ON TIME$="10:00:00" GOSUB 1000
20 TIME$ ON
.
.
1000 BEEP : FOR I=1 TO 10 : NEXT I : BEEP
1010 CLS
1020 PRINT "Aren't you supposed to be somewhere
right now?"
1030 PRINT "(Press any key to continue program)"
1040 A$ = INKEY$
1050 IF A$ = "" THEN GOTO 1040
1060 RETURN
```

This program runs as normal until the clock reaches ten o'clock, at which time BASIC drops whatever it is doing and jumps to line 1000.

The STOP command is useful for *masking* an interrupt. For example, you may have both TIME$ and MDM interrupts defined. However, you want the modem interrupt to take precedence over the time interrupt (that is, you want to receive all incoming data before servicing the TIME$ interrupt).

By beginning the MDM interrupt routine with a TIME$ STOP, you prevent the TIME$ interrupt, while still keeping track of any TIME$ condition. At the end of the routine, re-enable with TIME$ ON. If a TIME$ condition occurred while processing the MDM interrupt, BASIC then branches to the TIME$ interrupt routine.

Your program may look like:

```
10 ON TIME$ = "11:30:00" GOSUB 2000
20 ON MDM GOSUB 2000
30 TIME$ ON
40 MDM ON
.
.
.
1000 'MDM Servicing Routine
1010 TIME$ STOP     'Mask TIME$ interrupt
.
.
.
1090 TIME$ ON       'Unmask TIME$ interrupt
1100 RETURN
2000 'TIME$ Servicing Routine
.
.
```

You can find a description of the exact syntax of each command under **“BASIC Keywords.”**

15 / Input and Output

Your Model 100 is a very versatile computer in terms of input and output. It allows you to send and receive data on several *devices*. A device is a unit which has capabilities to store, display, or transmit data. Devices on your Model 100 include:

- LCD Screen
- Keyboard
- Line Printer
- Sound Generator
- CPU Ports
- RAM (Random Access Memory)
- Cassette Tape Recorder
- Telephone Modem
- RS-232C Communications Port

Model 100 BASIC lets you access each of these devices via simple commands. The BASIC commands fall into two broad categories, *immediate* I/O and *file* I/O.

Immediate I/O consists of I/O which you, as a user, have direct contact with. This category includes the screen, the keyboard, the printer, the sound generator, and the CPU ports.

File I/O deals with data which the Model 100 stores or transmits internally with no direct contact to you. This category includes RAM files, cassette files, modem files, communication files, LCD files, and printer files. The first two file types are stored files and the latter four are transmitted files.

Immediate I/O commands are different for each device and will be discussed by device. Many of the file I/O commands are ''generic'' in that the commands are very similar, regardless of the device type. These commands will be discussed by command.

(**Note:** For a complete discussion of each command, see **''BASIC Keywords.''**)

Screen I/O

Your Model 100's LCD Screen has 320 positions (40 columns X 8 lines) at which you can display characters. The screen can also be divided into 15,360 individual dots (or *pixels*) which you can turn ''on'' and ''off.''

By default, whenever BASIC gets to the end of the Screen, it *scrolls* the text. Scrolling erases the line at the top of the Screen and moves all lines up one line. If BASIC must scroll to print a line, then all of the non-text pixels are erased.

The Screen I/O commands are shown in Table 15-1.

Keyword	Operation
CLS	Clears the screen of all LCD dots.
CSRLIN	Returns the current vertical position of the cursor.
LCOPY	Copies the text on the screen to the printer.
LINE	Draws a line from the point on the screen to another.
LIST	Lists the current program
POS	Returns the current horizontal position of the cursor.
PRESET	Turns off a pixel at a specified row and column.
PRINT	Prints data at the current cursor position.
PRINT @	Prints data at a specified screen position.
PRINT USING	Prints formatted data on the screen.
PSET	Turns on a pixel at a specified row and column.
SCREEN	Turns on or off the Function Key labels.
TAB	Prints data at a specified horizontal screen position

Table 15-1

Keyboard Input

BASIC lets you take full advantage of your Model 100's keyboard, using two input commands, and two input functions.

Keyword	Operation
INPUT	Accepts data from the keyboard.
INPUT$	Returns a given number of characters typed in from the keyboard.
INKEY$	Returns the string value of any key currently pressed, if any.
LINE INPUT	Accepts a string of characters from the keyboard.
ON KEY GOSUB	Defines a Function Key interrupt.

Table 15-2

Printer I/O

BASIC has four commands which send data to the printer, and one function which returns information from the printer.

Keyword	Operation
LCOPY	Copies the text on the screen onto the printer.
LLIST	Prints out the current program.
LPRINT	Prints data onto the printer.
LPRINT USING	Prints formatted data onto the printer.
LPOS	Returns the current horizontal position of the cursor.
TAB	Prints data at a specified horizontal screen position.

Table 15-3

Sound Generator

Your Model 100 has a built-in sound generator, capable of producing sounds over five octaves. BASIC provides four commands which access the sound generator.

Keyword	Operation
BEEP	Causes the sound generator to "beep" for about 1/2 second.
SOUND	Causes the sound generator to produce a tone with a given pitch and length.
SOUND ON	Produces a tone whenever: a) the Model 100 is loading from the cassette; or b) the Model 100 is awaiting a carrier signal from the modem.
SOUND OFF	Nulifies SOUND ON.

Table 15-4

CPU Ports

The Model 100 uses an 80C85 central processing unit (''CPU''). This CPU has 256 I/O ''ports'' through which it communicates with the peripheral devices (the screen, the keyboard, and so on). The Model 100 operating system provides the necessary programs which tell the CPU how to use these ports, so normally, you needn't worry about them. However, in some applications, you may find it necessary to access these ports, via the following command and function.

Keyword	Operation
OUT	Outputs a byte to a port.
INP	Inputs a byte from a port.

Table 15-5

Cassette

Many times you'll want to store long programs and data files on cassette tape. The commands for accessing the cassette player are listed in Table 15-6.

Keyword	Operation
CLOAD	Loads a BASIC program from cassette tape.
CLOAD?	Verifies a cassette load.
CLOADM	Loads a machine language program from cassette tape.
CSAVE	Writes a BASIC program to cassette tape.
CSAVEM	Writes a machine language program to cassette tape.
MOTOR	Turns the cassette player motor on or off.

Table 15-6

File I/O

Treating each of the I/O devices as a ''file'' gives you the advantages of flexible code. This is because many of the file commands reference only a *file number*, given to the file via the OPEN statement. By changing OPEN statements, you can change a great many operations of your program without changing many lines of code.

Files can be *program files* or they can be *data files*. Model 100 BASIC lets you access files on six devices: RAM, cassette tape, RS-232C communications lines, telephone modem lines, printers, and the LCD screen. The abbreviations for the devices are listed in Table 15-7.

Abbreviation	Device
CAS	Cassette Tape File
COM	RS-232C Communications File
LCD	Screen File
LPT	Line Printer File
MDM	Telephone Modem File
RAM	RAM File

Table 15-7

Note that files on different devices have different forms and uses. RAM and CAS files constitute actual stored files. COM, LCD, LPT, and MDM are *transmission* files, that is, they involve sending data to and from devices such as the printer or LCD screen.

Model 100 BASIC has several commands which let you manipulate your files. To specify the device where the file is, you include the device abbreviation followed either by the *filename* or the *transmission configuration*.

RAM Files

You must specify the file name. This is a string of one to six characters, the first of which must be a letter, followed by a period and a two character *extension*. BASIC files use the extension **BA**, data and text files use the extension **DO**, and machine-language programs use the extension **CO**. Often the extension and even the word **RAM** are optional — BASIC assumes RAM and the extension from the context of the command.

For example,

"RAM:ACCTS.BA" **"RAM:MEMTST.CO"** **"NOTES.DO"**

Cassette Files

Like RAM files, you must specify a file name consisting of a string of one to six characters, the first of which must be letter. Unlike RAM files, you needn't specify any extension, and some commands even let you omit the file name, in which case BASIC uses the first cassette file it comes to.

For example,

"CAS:DATA1" **"CAS:TIMSET"** **"CAS:"**

RS-232C Files (COM)

You must specify the transmission configuration, which consists of a five character string of the pattern rwpbs. See Table 15-8.

r	**Baud Rate** This is a number from 1 to 9, where 1 = 75; 2 = 110; 3 = 300; 4 = 600; 5 = 1200; 6 = 2400; 7 = 4800; 8 = 9600; 9 = 19200.
w	**Word Size** This is a number from 6 to 8, where 6 = 6 bits; 7 = 7 bits; 8 = 8 bits.
p	**Parity** Either E,O,I, or N, where E = Even; O = Odd; I = Ignore; N = None.
b	**Stop Bits** Either 1 or 2, where 1 = 1 stop bit; 2 = 2 stop bits.
s	**XON/XOFF Status** Either E or D, where E = Enable; D = Disable. When enabled, either the Model 100 or the computer on the other end of the RS-232C line can automatically transmit an XON or XOFF to start or stop transmission. When disabled, any XON or XOFF must either come from the program or by pressing CTRL Q or CTRL S, respectively.

Table 15-8

For example,

"COM:78N1D" **"COM:96O2E"**

Modem Files (MDM)

You must specify the transmission configuration which consists of a four character string of the pattern ***wpbs***, as defined above for RS-232C files. (BASIC automatically sets the baud rate to 300 baud.)

For example,

"MDM:8N2E" **"MDM:6E1D"**

Screen Files (LCD) and Printer Files (LPT)

You needn't specify a file name or a configuration. For example,

"LCD:" **"LPT:"**

The file I/O commands and functions are listed in Table 15-8.

Keyword	Operation
CLOSE	Closes a file.
EOF	Function returning end of file condition.
INPUT$	Function returning a specified number of characters from a file.
INPUT	Reads data from a file.
KILL	Deletes a file.
LINE INPUT	Reads a string of characters terminated by a carriage return.
LOAD	Loads the BASIC program file into memory.
LOADM	Loads a machine language program into memory.
MAXFILES	Specifies the maximum number of files your program may have open at one time.
MERGE	Merges a BASIC program file with the program currently in BASIC memory.
NAME	Changes the name of a file.
OPEN	Allocates a buffer (a temporary block of memory) for I/O to the device.
PRINT	Send data to a file.
PRINT USING	Send formatted data into a file.
RUN	Begins execution of a program.
RUNM	Loads and runs a machine language program.
SAVE	Sends a BASIC program to a device.
SAVEM	Sends a machine language program to a device.

Table 15-8

(**Note:** All commands and functions are NOT valid for all devices, for example, you can't load a program from the printer! Refer to **"BASIC Keywords,"** for details on each command.)

16 / BASIC Keywords

ATN
Arctangent

ATN(*numeric expression*)

This function returns the arctangent of numeric expression (in radians). The resulting value ranges from - π to π.

Example

```
ARC = ATN(TH)
```

If TH contains the value 0.5, then this statement sets ARC equal to 0.46364760900081.

BEEP
Emit a Tone

BEEP

This command causes the sound generator to emit a tone for approximately 1/2 second.

Example

```
10 BEEP
```

emits a "beep."

CALL
Call a Machine Level Subroutine

CALL *address, expression1, expression2*

Calls a machine level subroutine beginning at *address*. Upon entry to the subroutine, the A register contains the value of *expression1* and the HL register contains the value of *expression2*. *expression1* is optional and may range from 0 to 255. *expression2* is also optional and may range from −32768 to 65535.

Example

```
100 CALL 60000,10,VARPTR(A%)
```

calls a subroutine beginning at address 60000. Upon entry to the subroutine, register A contains 10, and register HL contains the address of the variable A%

CDBL
Convert to Double-Precision

CDBL(*numeric expression*)

This function converts the value of *numeric expression* to a double-precision number according to the conversion rules given under **"Data Types and Data Manipulation."**

Example

```
10 A# = CDBL(A%)
```

if A% contains 34, then A# contains 34.0000000000000.

CHR$
Character Value

CHR$(*numeric expression*)

The CHR$ function returns the ASCII character for the value of *numeric expression*. *numeric expression* must lie in the range of 0 to 255. CHR$ is the inverse of the function ASC. For a list of ASCII codes, see the Appendices.

Examples

```
PRINT CHR$(65)
```

prints the character A.

```
PRINT "He said ";CHR$(34);"Hello";CHR$(34)
```

prints the message

```
He said "Hello"
```

(**Note:** 34 is the ASCII code for the double quote mark, ".)

CINT
Convert to Integer

CINT(*numeric expression*)

CINT truncates the decimal portion of *numeric expression*. The resulting value must lie in the range −32768 to 32767.

Examples

```
10 A% = CINT(45.67)
```

sets A% equal to 45.

```
10 A% = CINT(-2343.55823)
```

sets A% equal to −2343.

CLEAR
Clear Variables and Allocate String Space

CLEAR *string space, high memory*

This command clears the values in all numeric and string variables, and closes all open files. *string space* is a numeric expression and is optional. If present, BASIC allocates the specified number of bytes of memory for string storage. If you omit *string space*, BASIC allocates 256 bytes by default.

high memory is a numeric expression and is also optional. If present, BASIC sets the top of its memory at the given value. You may use the word MAXRAM to specify the maximum value available RAM (dependent on the configuration of your Model 100). If *high memory* isn't used, then BASIC defaults to MAXRAM.

If you intend on using machine-language subroutines, you need to use CLEAR to protect high memory space for the subroutines.

Note that when you enter BASIC, it always resets the *string space* allocation to 256 bytes. However, if you have protected a portion of *high memory* in a previous program, BASIC keeps this part of memory protected until you CLEAR it.

Examples

```
10 CLEAR
```

clears all variables, closes open files, sets the available string space to 256 bytes, and releases all of available memory to BASIC:

```
CLEAR 100,50000
```

clears all variables, closes open files, sets the available string space to 100 bytes, and set 50000 as the highest memory address available to BASIC.

CLOAD
Load a Program From Cassette

CLOAD *"filename"*, **R**

The CLOAD command clears the current BASIC program and loads a BASIC program from cassette tape. *filename* consists of a string of one to six characters, the first of which is a letter.

filename is optional; if used, BASIC searches the tape until it finds *filename*. If not used, BASIC loads the first BASIC program it finds. **R** is also optional; if used, BASIC executes the new program as soon as the load is complete.

As BASIC searches the tape, it prints out the names of any files it skips over.

(**Note:** CLOAD is identical to LOAD "CAS:)

Examples

```
CLOAD "ACCT",R
```

loads and runs the BASIC program ACCT stored on cassette tape.

```
CLOAD
```

loads the first BASIC program found on the cassette tape.

CLOAD?
Verify a Cassette Load

CLOAD? *filename*

This command compares *filename* with the BASIC program currently in memory. If there are any differences, BASIC displays the message `Verify failed` on the Screen; otherwise, BASIC simply prints `Ok`.

As with CLOAD, BASIC prints out the names of any files CLOAD? skips over.

Example

```
CLOAD? "ACCT"
```

compares the cassette file ACCT with the program currently in memory.

CLOADM
Load a Machine-Language Program From Cassette

CLOADM *"filename"*

The CLOADM command loads the program called *filename* from cassette tape into memory, at the address specified when it was written to the cassette tape. *filename* consists of one to six characters, the first of which must be a letter. It is optional; if omitted, BASIC loads the first machine-language program it finds on the tape.

As BASIC searches the tape, it prints out the names of any files it skips over.

(**Note:** This command functions identically to LOADM "CAS:)

Example

```
CLOADM "MEMTST"
```

loads the machine program MEMTST from the cassette.

CLOSE
Close Open Files

CLOSE *file number list*

This command closes the files specified in *file number list*. *file number* is the number under which the file was opened (See **OPEN**), and is optional; if omitted, BASIC closes all open files.

Example

```
CLOSE 1,2,3
```

closes the files associated with file numbers 1, 2, and 3.

CLS
Clear Screen

CLS

CLS turns off all of the LCD pixels on the Screen and moves the Cursor to the upper-left corner of the Screen.

Example

```
100 CLS: PRINT "The old screen is gone!"
```

This clears the Screen and prints out the message in the upper-left corner.

COM ON/OFF/STOP
Enable/Disable Communications Interrupt

COM ON or **OFF** or **STOP**

This enables or disables the ON COM interrupt. ON enables the interrupt so that if a character is received via the RS-232C port, BASIC jumps to the subroutine defined in the ON COM command. OFF disables the interrupt. STOP disables the interrupt. However, BASIC "remembers" that a character was received, so that if you issue a COM ON command, BASIC jumps immediately to the interrupt subroutine.

Examples

```
10 COM ON
```

enables the communications interrupt.

CONT
Continue Execution

CONT

CONT resumes execution of a program after you have pressed **BREAK** or after BASIC has encountered a STOP command in the program.

This command is primarily for debugging purposes. After you press **BREAK** or BASIC encounters a STOP command, you can examine any of the variables with the PRINT command, as well as change variable values. To continue, simply type **CONT** and press **ENTER**.

(**Note:** You cannot resume execution if you change any of the lines of the program, either through editing, deletion, or insertion.

Example

```
CONT
```

resumes execution of the BASIC program.

COS
Cosine

COS(*numeric expression)*

This function returns the cosine of angle given by *numeric expression*. You must give this angle in radians.

Example

```
10 Y = COS(60*0.01745329)
```

assigns Y the value 0.50000013093981.

80 27/53

CSAVE
Save a Program on Cassette

CSAVE *"filename"*,**A**

CSAVE stores the current BASIC program on cassette tape. *filename* consists of a string of one to six characters, the first of which is a letter. **A** is optional; if used, BASIC saves the program in ASCII format. If omitted, BASIC stores the program in a compressed form.

To perform certain commands, such as MERGE, programs must be stored in ASCII format. However, for most uses, you'll want your programs stored in a compressed format to save space on the tape.

(**Note:** This command functions identically to **SAVE "CAS:"**)

```
CSAVE "TANDY"
```

This example saves the current program onto cassette tape (in compressed format) under the name "TANDY."

```
CSAVE "TANDY",A
```

This line saves the current program onto cassette tape (in ASCII format) under the name "TANDY."

CSAVEM
Save a Machine Language Program to Cassette

CSAVEM *"filename"*,*start address, end address, entry address*

CSAVEM writes the program stored from *start address* to *end address* to cassette tape, under the name *filename*. *filename* consists of a string of one to six characters, the first of which is a letter. *entry address* is optional; if omitted, BASIC assumes that the program entry address is the same as the start address.

(**Note:** This command functions identically to **SAVEM "CAS:"**)

Example

```
CSAVEM "MEMTST",50000,50305,50020
```

This line writes the program stored from addresses 50000 to 50305 with the entry point at 50020 onto cassette tape, giving the file the name "MEMTST."

CSNG
Convert to Single Precision

CSNG(*numeric expression***)**

CSNG returns the single-precision form of *numeric expression,* according to the conversion rules listed under "**Data Conversion.**"

Example

```
10 A! = CSNG(0.66666666666)
```

sets A! equal to 0.666667.

CSRLIN
Vertical Cursor Position

CSRLIN

This function returns the vertical position (line number) of the Cursor, where 0 is the top line and 7 is the bottom line.

Example

```
10 CLS: A% = CSRLIN
```

clears the Screen and assigns A% the value 0.

DATA
Define a Data Set

DATA *constant list*

DATA defines a set of constants (numeric and/or string) to be accessed by a READ command elsewhere in the program. See also **READ** and **RESTORE**.

Example

```
DATA 10,25,50,15,"Probabilities","Total"
```

stores the given values.

DATE$
Current Date

DATE$ keeps track of the current date, in string form. You may access it like any string variable, including setting a new date. Date string has the form **MM/DD/YY** where $01 \leq MM \leq 13$, $01 \leq DD \leq 39$, and $00 \leq YY \leq 99$. BASIC automatically updates DATE$ when the clock changes from 23:59:59 to 00:00:00.

Examples

```
PRINT DATE$
```

prints the current date.

```
DATE$ = "11/02/82"
```

sets the current date to November 11, 1982.

DAY$
Current Day of Week

This string variable keeps track of the current day of the week, in string form. You may access DAY$ like any string variable, including setting a new day. DAY$ consists of a three letter string made up of the first three letters of the day name (for example, **Thu** signifies **Thursday**). BASIC automatically updates DAY$ when the clock changes from 23:59:59 to 00:00:00.

Examples

```
PRINT DAY$
```

prints the current day of the week.

```
DAY$ = "Fri
```

sets the current day as Friday. (Note that BASIC doesn't care whether the DAY$ string is in upper- or lowercase — it automatically converts the string to one uppercase letter followed by two lowercase letters.) Valid strings include:

Mon   Tue   Wed   Thu   Fri   Sat   Sun

and all upper- and lowercase combinations of each.

DEFDBL
Define Double-Precision Variables

DEFDBL *letter list*

DEFDBL defines all of the variables which begin with the letters in *letter list* to be double-precision variables. *letter list* consist of individual letters and/or letter ranges of the form *letter1 - letter2*.

Example

```
100 DEFDBL D, X-Z
```

defines as double-precision all variables beginning with the letters D, X, Y, and Z. Note that declaration tags override **DEFDBL**. For example, given the above DEFDBL, the variable D$ and Z% are string type and integer type, respectively.

DEFINT
Define Integer Variables

DEFINT *letter list*

This command defines all of the variables which begin with the letters in *letter list* to be integer variables. *letter list* consist of individual letters and/or letter ranges of the form *letter1 - letter2*.

Example

```
120 DEFINT D, X-Z
```

defines as integer type all variables beginning with the letters D, X, Y, and Z. Note that declaration tags override DEFINT. For example, given the above DEFINT, the variable D$ and Z! are string and single precision type, respectively.

DEFSNG
Define Single-Precision Variables

DEFSNG *letter list*

DEFSNG defines all of the variables which begin with the letters in *letter list* to be single-precision variables. letter list consist of individual letters and/or letter ranges of the form *letter1 - letter2*.

Example

```
30 DEFSNG D,X-Z
```

defines as single-precision type all variables beginning with the letters D, X, Y, and Z. Note that declaration tags override **DEFSNG**. For example, given the above DEFSNG, the variable D$ and Z# are string and double precision type, respectively.

DEFSTR
Define String Variables

DEFSTR *letter list*

This command defines all of the variables which begin with the letters in *letter list* to be string variables. letter list consist of individual letters and/or letter ranges of the form *letter1 - letter2*.

Example

```
100 DEFSTR D, X-Z
```

defines as integer type all variables beginning with the letters D, X, Y, and Z. Note that declaration tags override **DEFSTR**. For example, given the above DEFINT, the variable D% and Z! are integer and single precision type, respectively.

DIM
Define Array Size

DIM *variable name(dimensions) list*

DIM defines *variable name* as an array with the given dimensions. *dimensions* is a list of one or more numeric expressions, defining the "height," "width," and so on for the array. The expressions must evaluate to positive values. Since BASIC arrays begin with the "zeroth" element, the actual size in any dimension is one plus the given dimension.

The number of dimensions you may list depends only on the amount of available memory. To redimension an array, you must first use the command **CLEAR** (this destroys all variable values).

Examples

```
DIM A$(10), BAL%(10,10)
```

defines a string array, A$, which consists of 11 elements, A$(0) through A$(10), and an integer array, BAL%, which consists of 121 elements, BAL%(0,0) through BAL%(10,10).

EDIT
Edit a BASIC Program

EDIT *line number range*

EDIT enters the text editor using the lines given by *line number range*. line number range may be:

null	edit the entire program.
line1-line2	edit from *line1* to *line2*, inclusive.
-line2	edit from beginning of the program to *line2*.
line1-	edit from *line1* to the end of the program.
.	edit last accessed line number (last edited, entered, listed, and so on).

To exit Edit, press (F8). Note that while in the Edit Mode, you may insert and delete lines. However, if you insert a line which already exists in the program, the new line replaces the old line.

If you edit a line in such a way that it is not a valid program line (for example, longer than 255 characters, no line number, invalid line number), the Editor tells you

```
Text Ill-formed
Press space bar for TEXT
```

Pressing the space bar returns you to the Text Editor. For more information on the Text Editor, see Part II of this manual.

Examples

```
EDIT
```

lets you edit the entire program.

```
EDIT 100-500
```

lets you edit lines 100 through 500.

```
EDIT 100-
```

lets you edit from line 100 to the end of the program.

END
End Execution

END

END terminates execution of the BASIC program. END statements may be placed anywhere in the program. If omitted, BASIC executes up to the physical end of the program.

Example

```
10 INPUT S1, S2
20 GOSUB 100
.
.
.
90 END
100 H=SQR(S1*S1+S2*S2)
110 RETURN
```

The END statement in line 90 prevents BASIC from entering the subroutine in line 100 from anywhere but a GOSUB call.

EOF
Test for End-of-File

EOF *(file number)*

EOF tests for an end-of-file condition on RAM, cassette, or communications files. *file number* is the buffer number assigned when the file was OPEN'ed. The function returns a ''logical'' answer, either ''true'' (-1) if you have reached the end of the file, or else ''false'' (0) if you have not reached the end of the file.

Example

```
100 IF EOF(1) THEN 200
```

checks the file assigned to buffer 1 for end of file. If true, then the program jumps to line 200.

ERL
Get Line Number of Error

```
ERL
```

ERL returns the line number of the last error. If the last error which occured was not in a program line but from a direct mode command, ERL returns the value 65535. This command is useful in conjunction with the ON ERROR GOTO command.

Example

```
100 ON ERROR GOTO 2000
.
.
.
2000 IF ERR = 23 THEN RESUME ELSE PRINT "Error";ERR;
     "in line";ERL: STOP
```

If an error occurs in the program, execution jumps to line 2000. If the error was an I/O error (ERR = 23), then BASIC simply retries the I/O (RESUME). If there was some other type of error, say a syntax error in line 1000, then BASIC displays the message:

```
Error 2 in line 1000
```

and stops the program. See also ON ERROR and ERR.

ERR
Get Error Code Number

```
ERR
```

This function returns the error code number of the last error. This command is useful in conjunction with the ON ERROR GOTO command.

Example

```
100 ON ERROR GOTO 2000
.
.
.
2000 IF ERR = 18 THEN PRINT "I/O Error " ELSE STOP
2010 INPUT "Continue(Y/N)"; A$
2020 IF A$ = "Y" THEN RESUME ELSE STOP
```

If an error occurs in the program, then BASIC jumps to line 2000. If the error was an I/O error (error 18), then BASIC prints out the message and prompt and waits on your response

ERROR
Simulate an Error

ERROR *numeric expression*

This command simulates the error specified by numeric expression. BASIC behaves just as if your program had committed the error.

Example

```
ERROR 4
```

prints OD Error.

```
100 ERROR 10
```

prints DD Error in 100 and stops execution of the program.

EXP
Exponential (Antilog)

EXP(*numeric expression*)

EXP returns the exponential (or ''natural'' antilog) of *numeric expression. numeric expression* must be in the range ± 87.3365 or an overflow error occurs. EXP is the opposite of the function LOG.

Example

```
PRINT EXP(14)
```

prints 1202604.2841644, the natural antilog of 14.

FILES
Display File Names

FILES

This command will cause BASIC to display all of the files currently stored in RAM without exiting BASIC.

Example

```
FILES
```

FIX
Truncate Real Numbers

FIX *(numeric expression)*

FIX returns the whole number portion of *numeric expression.*

(**Note:** The difference between FIX and INT is that for negative numbers, FIX simply truncates numeric expression while INT returns the whole number not greater than numeric expression.

Examples

```
10 A = FIX(1440.43)
```

sets A equal to 1440.

```
10 A = FIX(-33494123.4442)
```

sets A equal to -33494123.

FOR...NEXT
Establish Program Looping

FOR *counter variable = initial value* **TO** *final value* **STEP** *increment*
.
.
.
NEXT *counter variable*

These commands execute the statements between the FOR and NEXT loop repetitively, varying *counter variable* from *initial value* to *final value*, adding *increment* to it each time BASIC ends the loop. *initial value, final value*, and *increment* are all numeric expressions. STEP *increment* is optional; if omitted, BASIC assumes STEP 1.

BASIC executes the loop at least once, even if *variable* exceeds *final value* the first time the FOR statement is read. Also, BASIC evaluates *initial value, final value*, and *increment* the first time it executes the line. If these expressions are variables, changing the values of these variables later in the loop has no effect on the execution of the loop.

You may ''nest'' FOR...NEXT loops with other FOR...NEXT loops. The number of nested loops is dependent only on the size of remaining memory. Note that all inner loops must be contained entirely within the next outer loop. For example:

```
10 FOR I=1 TO 100        10 FOR I=1 TO 100
20 FOR J=I TO 20         20 FOR J=1 TO 20
30 B(I,J) = A(I,J)       30 B(I,J) = A(I,J)
40 NEXT J                40 NEXT I
50 NEXT I                50 NEXT J
```

Legal **Illegal**

Examples

```
10 FOR I=10 TO 1 STEP -1
20 PRINT I;
30 NEXT
```

prints the numbers 10 through 1. (Note that you may use negative values for increment.)

```
10 FOR K=B TO E
20 PRINT K
30 NEXT
```

If B equals 4 and E equals 2, this routine prints out the number 4. Since no STEP is specified, STEP 1 is assumed. BASIC then increments K by 1 to the value 5. Since 5 is greater than 2, the loop ends.

```
10 FOR I=1 TO 3
20 PRINT "OUTER LOOP"
30 FOR J=1 TO 2
40 PRINT "      INNER LOOP"
50 NEXT J
60 NEXT I
```

prints:

```
OUTER LOOP
      INNER LOOP
      INNER LOOP
OUTER LOOP
      INNER LOOP
      INNER LOOP
OUTER LOOP
      INNER LOOP
      INNER LOOP
```

Note that lines 50 and 60 could be condensed to NEXT J,I (but not NEXT I,J). You may also simply say NEXT and BASIC will ''know'' which loop it is in.

FRE
Free Memory Space

FRE*(dummy expression)*

FRE returns the current amount of unused numeric memory in bytes when *dummy expression* is numeric and the current total amount of unused string space when *dummy expression* is string type.

Examples

```
PRINT FRE(0)
```

prints out the current free numeric memory space.

```
?FRE("")
```

prints out the current free string space.

GOSUB
Call a BASIC Subroutine

GOSUB *line number*

This command transfers program control to the subroutine beginning at *line number*. When BASIC encounters a RETURN command in the subroutine, it jumps back to the command immediately following the GOSUB. You must always terminate a subroutine with a RETURN command.

Example

```
100 GOSUB 1000
110 PRINT "Average =";AV
.
.
.
990 END
1000 'Averaging Subroutine
1010 FOR I=1 TO 20
1020 SM = A(I)
1030 NEXT I
1040 AV = SM / 20
1050 RETURN
```

Line 100 calls the subroutine at line 1000. BASIC executes lines 1000 through 1040, and then jumps back to line 110 and begins execution there.

GOTO
Branch Program Execution

GOTO *line number*

GOTO branches program control to the specified *line number*. Used alone, GOTO *line number* results in an ''unconditional'' (or automatic) branch. You may also use GOTO in conjunction with conditional expressions, such as IF and ON ERROR. This results in ''conditional'' branching.

You can use GOTO in the Immediate Mode to cause execution to begin at the specified line number, without an automatic CLEAR. This allows you to enter a program at a specific point, without destroying any old variables (the command RUN tells BASIC to first clear all of memory before beginning execution).

Examples

```
200 GOTO 10
```

branches control unconditional to line 10.

```
100 IF AN$ = "Y" GOTO 1000 ELSE GOTO 2000
```

if AN$ equals "Y," then BASIC branches to line 1000; otherwise BASIC branches to line 2000.

```
A=1.32 : GOTO 1000
```

assigns A the value 1.32 and begins execution at line 1000.

HIMEM
Get High Memory Address

HIMEM

This function returns the top address of memory available to BASIC. You may change this value with the CLEAR command.

Example

```
PRINT HIMEM
```

prints the current top address of BASIC memory.

IF...THEN...ELSE
Test Relational Expression

IF *relational* or *logical expression* **THEN** *command(s)1*
ELSE *command(s)2*

The IF/THEN statements test the logical "truth" of relational or logical expression (relational and logical expressions are defined under **"Expressions,"** earlier in this section). If the expression is "true," then BASIC executes *command(s)1*. If the expression is "false," BASIC executes *command(s)2*.

If THEN *command(s)1* is a THEN GOTO line number, BASIC also accepts THEN line number and GOTO line number as equivalent terms.

ELSE *command(s)2* is optional; if omitted, BASIC assumes the ELSE clause is the next line. If ELSE *command(s)2* is a GOTO line number, then ELSE line number is an equivalent term.

In numeric terms, "false" has the value zero, and "true" is any value except zero.

Example

```
10 IF A < 100 THEN 100
20
```

tests the condition A < 100 — if true, then BASIC jumps to line 100; if false, then BASIC continues execution at line 20.

```
10 IF A = 10 OR A = 20 THEN B$ = "Paid" ELSE B$ =
"Not Paid"
```

tests the condition A = 10 OR A = 20 — if true, then BASIC assigns B$ the string "Paid"; if false, then BASIC assigns B$ the string "Not Paid".

INKEY$
Poll Keyboard

INKEY$

This function returns the string value of the key currently pressed, if any. If no key is pressed, the function returns a null character (""). In either case, BASIC doesn't wait for keyboard input, but goes to the next statement.

(**Note:** If you press an undefined Function Key, PASTE or LABEL, INKEY$ returns an ASCII 0 with a length of 1.)

Example

```
10 A$ = INKEY$
20 IF A$ = "" THEN 10
30 .
.
.
```

sets A$ equal to the string value of any key pressed. If you haven't pressed a key, then A$ contains the null character ("") and BASIC jumps back to line 10. If you have pressed a key, A$ contains the character representation of the key you pressed, and hence BASIC continues execution at line 30.

INP
Input From a Port

INP *(port number)*

INP returns a byte from the specified port. *port number* must be a numeric expression in the range of 0 to 255. INP is the complement function to the OUT command.

Example

```
A% = INP(5)
```

sets A% equal to the byte value at port 5.

INPUT
Input Data From Keyboard

INPUT *"prompt";variable list*

INPUT stops execution of your program until you enter data from the keyboard. The values you enter must be constants and must correspond in both number and type to the variables in *variable list*. *variable list* consists of any number of variable names, both string and numeric, separated by commas. The optional *"prompt"* is any valid string expression. BASIC displays *prompt* prior to accepting your input.

While BASIC is awaiting your input, it displays a question mark. At this point, you may enter enough data, separated by commas, for all of the variables in variable list, terminated with ENTER. Alternatively, you may enter each data item separately, pressing ENTER after each. In the latter case, after accepting the first value, BASIC displays two question marks as a prompt for subsequent input.

For string input, you may enclose the data in quotes, although BASIC doesn't require this. However, if the input string contains any leading blanks, commas, or colons, you must use quote marks. BASIC lets you input numeric data into string variables. (BASIC stores the ASCII value — not the numeric value!)

For numeric input, BASIC performs any necessary conversion to the numbers so that they fit into the variable. This conversion is identical to other data conversions in the program. If you attempt to input string data into a numeric variable, BASIC displays the message `?Redo from start` followed by another ? , and lets you try again. See "**Data Conversion**," earlier in this section for the data conversion rules.

Examples

```
10 INPUT "X and Y Coordinates";X,Y
```

BASIC displays:

```
X and Y Coordinates?
```

If you type 10,20 and press ENTER, then BASIC assigns X the value 20 and Y the value 30. If you type 10 and press ENTER, then BASIC assigns X the value 20, and then displays:

```
??
```

You may then type in the value for Y and press ENTER.

```
100 INPUT A$,B%,C$,D
```

This statement calls for you to input a string, a number, a string, and finally another number. BASIC prompts you with a ?. You may then type in:

```
Fort Worth,5641.321,Texas,76109 ENTER
```

This assigns "Fort Worth" to A$, 5641 to B% (note the conversion), "Texas" to C$, and 76109 to D. The following is equivalent:

```
"Fort Worth" ENTER
5641 ENTER
Texas ENTER
76109 ENTER
```

INPUT
Input From a File

INPUT # *file number, variable list*

This command inputs data sequentially from the file opened under *file number*. You may input data with this command from RAM, cassette tape, the RS-232C port, or the modem.

INPUT # functions identically to INPUT from the keyboard, except that the data comes from a file, and BASIC displays no question mark prompt. The data in the file must be separated by commas.

See also **OPEN**.

Example

```
10 INPUT #1,A$,B$,C
```

inputs values for A$, B$ and C from the file opened as file #1.

INPUT$
Input Characters From the Keyboard

INPUT$ *(numeric expression)*

This function returns a string of *numeric expression* characters from the keyboard. *numeric expression* must be in the range of 1 to 255. INPUT$ accepts all keys as input except BREAK. It does **not** echo (print on the Screen) your input.

Example

```
10 A$ = INPUT$(5)
```

waits for you to input five characters from the keyboard, and assigns the input string to A$.

INPUT$
Input Characters From a File

INPUT$ *(numeric expression, file number)*

This INPUT$ returns a string of a length given by *numeric expression* from the file opened under *file number*. *numeric expression* must be in the range of 1 to 255. INPUT$ accepts all keys as input except BREAK.

Example

```
10 A$ = INPUT$(5,1)
```

inputs five characters from the file opened as file #1, and assigns the input string to A$.

INSTR
Search a String

INSTR *(start position,search string,match string)*

INSTR searches *search string* for the first occurrence of *match string*, beginning at *start position*. If the *string* is found, INSTR returns the position in the string where it occurs. If *string* isn't found, then INSTR returns a zero.

start position is optional; if omitted INSTR starts the search at position one. Also, if *start position* is greater than the length of *search string*, INSTR returns a zero.

Example

```
PRINT INSTR("dimethylsulfate","sulfate")
```

prints 9 on the Screen (''sulfate'' begins at position 9 of ''dimethylsulfate'').

```
A% = INSTR(5,NM$,"Jim")
```

If NM$ contains the string ''JimBob'', then this line sets A% equal to 0 (''Jim'' does not occur past position 5 of ''JimBob'').

INT
Get Whole Number Representation

INT(*numeric expression*)

INT returns the whole number representation of *numeric expression*, not greater than *numeric expression*.

(**Note:** The difference between the functions INT and FIX is that for negative numbers, FIX simply truncates numeric expression while INT returns the whole number not greater than numeric expression.)

Examples

```
A# = INT(21444331113.443)
```

sets A# equal to 21444331113.

```
A# = INT(-214.995)
```

sets A# equal to −215.

IPL
Define Warm Start Program

IPL *''filename''*

IPL defines *filename* as the warm-startup program. *filename* is the name of a current RAM file. After executing this command, this program runs whenever you turn on your Model 100.

Example

```
IPL "TIMSET.BA"
```

Now whenever you turn on the Computer, it will execute the program TIMSET.BA. IPL will execute properly only if the Computer is turned off while in BASIC with the proper IPL program loaded.

KEY
Define Function Keys

KEY *function key number,string expression*

KEY defines *function key number* as *string expression*. *function key number* is a numeric expression from 1 to 8 and *string expression* must be 15 or less characters.

Example

```
KEY 6,"?TIME$" + CHR$(13)
```

defines Function Key 6 as ?TIME$ followed by a carriage return. Now whenever you press Function Key 6, BASIC returns the time. (Remember that "?" is an abbreviation for PRINT, and that ASCII character 13 is the code generated when you press **ENTER**.)

To reset the function keys to the cold start default, you must "call" two subroutines with the following commands:

```
CALL 23164,0,23366
CALL 27795
```

This resets the function keys to their original value.

KEY LIST
List Function Key Definitions

KEY LIST

Lists on the Screen the current definitions for the Function Keys, in the format:

```
key 1                    key 2
key 3                    key 4
key 5                    key 6
key 7                    key 8
```

Example

```
KEY LIST
```

Unless you have altered the function key definitions, BASIC displays:

```
Files                    Load"
Save                     Run
List

                         Menu
```

KEY ON/OFF/STOP
Enable/Disable Function Key Interrupts

KEY *(function key number)* **ON/OFF/STOP**

This statement enables or disables the function key interrupt. ON enables the interrupt so that if you press a Function Key, BASIC branches to the ON KEY subroutine. OFF disables the interrupt. STOP disables the interrupt, however, BASIC ''remembers'' that you pressed a Function Key, so that if you issue a KEY ON command, BASIC jumps immediately to the interrupt subroutine.

See **ON KEY GOSUB.**

Examples

```
100 KEY (2) ON
```

enables Function Key 2.

```
100 KEY ON
```

enables all Function Keys.

```
100 KEY (4) OFF
```

disables Function Key 4.

KILL
Delete a RAM File

KILL *''filename''*

KILL deletes a RAM file. *filename* is a string of one to six characters, the first of which must be a letter, plus a two character extension. **You *must* include the file's extension.**

If you have 200 bytes or less of free memory, KILL may not delete a file. If this situation occurs, delete program lines ''manually'' or go to TEXT, ''select'' a file, and put it in the PASTE Buffer. Then return to BASIC and KILL the unwanted files.

Example

```
KILL "BILLS.BA"
```

deletes the RAM file BILLS.BA.

LCOPY
Copy Screen to Printer

LCOPY

Prints the text on the Screen onto the printer. LCOPY ignores non-text data.

Example

```
LCOPY
```

prints the text on the Screen onto the printer.

LEFT$
Return Left Portion of a String

LEFT$*(string expression,length)*

LEFT returns the first *length* characters of *string expression*. *length* is a numeric expression.

Example

```
10 AC$ = LEFT$(PN$,3)
```

If PN$ contains the string "81755552161," then after execution of this command AC$ contains the string "817".

LEN
Get Size of a String

LEN*(string expression)*

LEN returns the number of characters in *string expression*.

Example

```
100 INPUT NM$
110 IF LEN(NM$) < 20 THEN NM$ = NM$ + " ": GOTO 110
.
.
```

adds spaces to the end of NM$ so that it is at least 20 characters long. This "left justifies" the input string, while "padding" on the right with spaces.

LET
Assignment Statement

LET *variable = expression*

This statement assigns value of *expression* to *variable*. *variable* must be of the same data type as *expression* (that is, numeric or string). For numeric expressions, BASIC performs any conversion necessary to fit expression into variable (see "**Data Conversion**," for the conversion rules).

LET is optional — it is included in Model 100 BASIC to be compatible with older forms of BASIC.

```
LET A$ = "The"
```

and

```
A$ = "The"
```

both assign the string "The" to A$.

LINE
Draw a Line on the Screen

LINE *(x1,y1) – (x2,y2), switch,* **BF**

LINE draws a line from coordinates *x1,y1* to *x2,y2*. *x1* and *x2* are numeric expressions which range from 0 to 239, and *y1* and *y2* are numeric expressions which range from 0 to 63. *(x1,y1)* is optional; if not used, BASIC starts the line from the *x,y* coordinates of the last LINE command, or from 0,0 if this is the first LINE command. You must always include the hyphen.

switch is a numeric expression and is optional; if used, odd values of switch tell BASIC to set the points of the line, and even values of switch tell BASIC to reset (that is, erase) the points on the line. If not present, BASIC assumes you mean to set the points of the line.

B tells BASIC to draw a box with corners at *(x1,y1)* and *(x2,y2)*. **BF** tells BASIC to fill in the box with *switch*. Both **B** and **BF** require that you specify *switch*.

Examples

```
10 LINE (20,20)-(50,63)
20 LINE -(30,30)
```

draws lines from (20,20) to (50,63), and from (50,63) to (30,30).

```
10 LINE (20,20)-(50,63),0
```

erases (resets) all points on a line from (20,20) to (50,63)

```
10 LINE (0,0)-(239,63),1,B
```

draws a box with corners at (0,0) and (239,63).

```
10 LINE (0,0)-(239,63),1,BF
```

draws a box with corners at (0,0) and (239,63) and then sets all of the points inside the box.

LIST
List Program on the Screen

LIST *line number range*

This command lists the line number range of the current program on the Screen. *line number range* may be:

null	lists the entire program.
line1-line2	lists from *line1* to *line2*, inclusive.
-line2	list from beginning of the program to *line 2*.
line1-	list from *line1* to the end of the program.
	lists the last accessed line number (last edited, entered, listed, and so on)

Example

```
LIST
```

displays the entire program.

```
LIST 100-300
```

displays from line 100 to line 300.

```
LIST .-
```

displays from the current line to the end of the program.

LLIST
List Program on the Printer

LLIST *line number range*

LLIST lists the *line number range* of the current program onto the printer. *line number range* may be:

null	lists the entire program.
line1-line2	lists from *line1* to *line2*, inclusive.
-line2	list from beginning of the program to *line2*.
line1-	list from *line1* to the end of the program.
	lists last accessed line number (last edited, entered, listed, and so on)

Example

```
LLIST
```

prints out the entire program.

```
LLIST 100-300
```

prints out line 100 to line 300.

```
LLIST -.
```

prints out from the first line of the program to the current line.

LINE INPUT
Input a String from the Keyboard

LINE INPUT *"prompt"; string vanable*

This statement stops execution of your program until you enter a string from the keyboard, then assigns that string to *string variable*. The optional *''prompt''* is any valid string constant which BASIC displays prior to accepting your input.

LINE INPUT differs from INPUT in that:

- BASIC doesn't display a question mark prompt
- You may have only one variable name
- BASIC assigns **all** input (including commas, leading blanks, and quote marks) to string variable

Example

```
10 LINE INPUT "Enter Name and Address:";NA$
```

displays:

```
Enter Name and Address:
```

and waits for you to enter data. If you typed:

```
John "Rocky" Smith, 5641 Lancaster, East Pearoe, Ohio ENTER
```

BASIC assigns NA$ the string John "Rocky" Smith, 5641 E. Lancaster, East Pearoe, Ohio.

LOAD
Load a BASIC Program

LOAD *"device:filename or configuration"*,**R**

LOAD loads a BASIC program from device. *filename* consists of a string of one to six characters, the first of which is a letter, *device* may be **RAM, CAS,COM,** or **MDM**. If *device* is **RAM**, then you may include the optional extension **.BA** or **.DO**. If *device* is **CAS**, then you use no extension.

For **COM**, configuration consists of a five character string of the pattern *rwpbs,* where

r	**Baud Rate** This is a number from 1 to 9, where 1 = 75; 2 = 110; 3 = 300; 4 = 600; 5 = 1200; 6 = 2400; 7 = 4800; 8 = 9600; 9 = 19200.
w	**Word Length** This is a number from 6 to 8, where 6 = 6 bits; 7 = 7 bits; 8 = 8 bits.
p	**Parity** Either E,O,I, or N, where E = Even; O = Odd; I = Ignore; N = None.
b	**Stop Bits** Either 1 or 2, where 1 = 1 stop bit; 2 = 2 stop bits.
s	**XON/XOFF Status** Either E or D, where E = Enable; D = Disable.

For **MDM**,configuration consists of a four character string of the pattern *wpbs,* defined above. (BASIC automatically sets the baud rate to 300 baud.)

R is optional; if used, BASIC runs the incoming program as soon as it has been read in.

Note that for **MDM** and **COM**, the person on the other end of the communications line must be ready to send the program using the same configurations, after you enter the LOAD command.

Cassette loads have several features not found in other load forms:

- You may omit *filespec,* in which case BASIC loads the BASIC program it finds.
- If *filespec* isn't the first program on the tape, BASIC prints the message `Skip:` followed by the name of the file it is skipping over.
- The command CLOAD *"filespec"* functions identically to LOAD **"CAS:***filespec"*.)

See also **SAVE**.

Examples

```
LOAD "RAM:TIMSET"
```

loads the BASIC program TIMSET.BA from memory.

```
LOAD "CAS:ACCT",R
```

loads and runs the BASIC program ACCT from cassette tape. (**Note:** The program could have been SAVE'd in either ASCII or binary format.)

```
LOAD "COM:78N1E"
```

loads a BASIC program from the RS-232C Communications Line, using 4800 baud, eight bit words, no parity, one stop bit, and line enable.

```
LOAD "MDM:7O2E",R
```

loads a BASIC program from the modem, using seven bit words, odd parity, two stop bits, and line enable.

LOADM
Load a Machine-Language Program

LOADM
“**RAM/CAS**:*filename*”

LOADM loads a machine-language program called *filename* from RAM or cassette tape, at the address specified when it was saved. *filename* consists of a string of one to six characters, the first of which is a letter. For a RAM file, you may optionally include the extension **.CO**. If you don't specify device, BASIC assumes **RAM**.

When BASIC loads the file, it prints out its start address, end address, and entry point, if any.

(**Note:** LOADM ‘‘CAS: functions identically to CLOADM.)

Examples

```
LOADM "MEMTST"
```

loads the machine-language program called MEMTST.CO from RAM.

```
LOADM "CAS:MEMTST"
```

loads the machine-language program called MEMTST from cassette tape.

LOG
Natural Logarithm

LOG *(numeric expression)*

LOG returns the natural logarithm (base ‘‘e’’) of *numeric expression*. *numeric expression* must be greater than zero.

Example

```
10 A = LOG(10)
```

sets A equal to 2.302585092994.

LPOS
Printer Column Position

LPOS *(dummy numeric expression)*

This command returns the current position of the printer print head within the printer buffer.

Example

```
LPRINT "Printer head position:";LPOS(0)
```

prints the message followed by the number.

LPRINT
Print Data on Printer

LPRINT *expression list*

LPRINT prints out the values of expression list on the printer. If the expressions are separated by commas, then the printer prints a value and advances to the next ''print zone'' before printing the next value. The print zones are defined every 14 columns (at column 0, column 14, column 28, and so on). If the expressions are separated by semicolons, the printer prints each value with no space between.

All numbers are printed with a trailing blank. If the number is negative, the sign precedes the number, otherwise a blank precedes the number. No blanks precede or follow strings.

(**Note:** You may not substitute L? for LPRINT.)

Examples

```
LPRINT "The total for ";A$;" was "; TT
```

If A$ contains the string ''April'' and TT contains the value 1332.44, this statement prints:

```
The total for April was 1332.44
```

```
LPRINT X,Y,Z
```

prints the value of X beginning in column 0, Y in column 14, and Z in column 28.

```
LPRINT X,,,Z
```

prints the value of X beginning in column 0, and Z in column 42 (two columns are skipped because of the two commas.)

LPRINT USING
Print Formatted Data on Printer

LPRINT USING *"format string";expression list*

This command formats and prints out the values of expression list using format string. For examples and a description of format string, see PRINT USING.

MAXFILES
Maximum Number of Files

MAXFILES contains the current maximum number of files. You may access MAXFILES like any numeric variable. By default, BASIC sets MAXFILES at 1.

Examples

```
10 MAXFILES = 5
```

sets the maximum number of open files to 5.

```
?MAXFILES
```

prints the current value of MAXFILES.

MAXRAM
Amount of Memory

MAXRAM contains the memory size of your Model 100. You may access it like any variable, except that you may not redefine its value.

Example

```
CLEAR 1000,MAXRAM
```

clears 1000 bytes for string storage and sets the high memory to the maximum amount for your machine.

MDM ON/OFF/STOP
Enable/Disable Modem Interrupt

MDM ON/OFF/STOP

This command enables or disables the ON MDM interrupt. ON enables the interrupt so that if a character is received via the modem, BASIC jumps to the subroutine defined in the ON MDM command. OFF disables the interrupt. STOP disables the interrupt, however, BASIC "remembers" that a character was received, so that if you issue a MDM ON command, BASIC jumps immediately to the interrupt subroutine.

Examples

```
10 MDM ON
```

enables the communications interrupt.

MENU
Return to Menu

MENU

MENU exits BASIC and returns you to the Model 100 Main Menu. If you were editing a current RAM file, BASIC rewrites the file before returning to the Menu (unless you entered NEW before entering MENU).

Example

```
1000 PRINT "Press Any Key to Return to Menu"
1010 A$ = INKEY$: IF A$ = "" GOTO 1010
1020 MENU
```

prints the message and returns to the Menu when you press any key.

MERGE
Merge Two Programs

MERGE *"device: filename or configuration"*

This command merges the lines from the **ASCII** formatted file called *filename* with the lines of the current program. If BASIC finds a duplicate line number, the line from *filename* replaces the current line.

device may be **RAM, CAS, COM,** or **MDM.** If you don't specify a device, BASIC assumes RAM. *filename* consists of a string of one to six characters, the first of which is a letter. For **RAM** files, you may include the optional extension. DO; if omitted, BASIC assumes that extension. unless there is Basic file of same name. For example if there are "PROG.DO" and "PROG.BA", "MERGE PROG" will cause to merge PROG.BA, and result error.

If *device* is **CAS**, then *filename* has no extension. *filename* is optional; if omitted, BASIC merges the first ASCII formatted cassette file it finds.

For **COM**, *configuration* consists of a five character string of the pattern *rwpbs*, where:

r	**Baud Rate** This is a number from 1 to 9, where 1=75; 2=110; 3=300; 4=600; 5=1200; 6=2400; 7=4800; 8=9600; 9=19200.
w	**Word Length** This is a number from 6 to 8, where 6=6 bits; 7=7 bits; 8=8 bits.
p	**Parity** Either E,O,I or N, where E=Even; O=Odd; I=Ignore; N=None.
b	**Stop Bits** Either 1 or 2, where 1=1 stop bit; 2=2 stop bits.
s	**XON/XOFF Status** Either E or D, where E=Enable; D=Disable.

For **MDM**, *configuration* consists of a four character string of the pattern *wpbs*, defined above. (BASIC automatically sets the baud rate to 300 baud.)

For information on storing files in ASCII format, see **SAVE** and **CSAVE**.

Examples

If the current program is:

```
10 FOR I=1 TO 100
20 PRINT AVE(I),BAL(I)
30 NEXT I
```

and the file ACT.DO contains the lines:

```
20 PRINT CD$,BAL(I)
25 PRINT PD$
40 MENU
```

then after the command

```
MERGE "RAM:ACT.DO"
```

the current program reads:

```
10 FOR I=1 TO 100
20 PRINT CD$,BAL(I)
25 PRINT PD$
30 NEXT I
40 MENU
```

Other examples:

```
MERGE "CAS:ACCT"
```

merges the cassette file ACCT with the program in memory.

```
MERGE "COM:78e1e"
```

merges the file coming in on the RS-232C line with the program in memory. (**Note:** The party on the other end of the RS-232C line must send the ASCII-oriented file using the same configuration, after you enter the MERGE command.)

```
MERGE "MDM:8e1e"
```

merges the file coming in on the modem with the program in memory. (**Note:** The party on the other end of the phone line must send the ASCII-oriented file using the same configuration, after you enter the MERGE command.)

MID$
Get Middle Characters of String

MID$*(string expression,position,length)*

This function returns *length* characters from *string* starting at *position*. *length* and *position* are numeric expressions. *length* is optional; if omitted, MID$ returns the entire portion of the *string* starting at *position*.

Example

```
10 HASH$ = MID$(A$,2,2)
```

If A$ contains the string "00349953," then this statement assigns string "34" to HASH$.

MID$
Replace Middle Characters of a String

MID$*(string expression1,position,length)* = *string expression2*

This MID$ replaces characters of *string expression1*, starting at *position*, with *string expression2*. *length* and *position* are numeric expressions. *length* is optional; if present it is ignored. *string expression1* always keeps its original size, even if it means truncating *string expression2* to fit.

Example

```
10 MID$(A$,5) = "FF"
```

If A$ contains the string "00000000," then this statement changes A$ to "0000FF00."

```
1000 MID$(A$,5) = "ABCDEF"
```

If A$ contains the string "00000000," then this statement changes A$ to "0000ABCD."

MOTOR
Turn Cassette Motor On and Off

MOTOR ON or **OFF**

MOTOR starts or stops the cassette recorder motor.

Example

```
MOTOR ON
```

starts the cassette recorder motor.

NAME...AS
Rename a RAM file

NAME "RAM:*old filename*" **AS "RAM:***new filename*"

This command renames *old filename* to *new filename*. *old filename* and *new filename* consist of strings of one to six characters, the first of which must be a letter, plus the two character extension. You must include the extension, and you may not change extensions. *old filename* must already exist and *new filename* must not already exist. **RAM** is optional.

Examples

```
NAME "ACCTS.DO" AS "OLDACT.DO"
```

renames the RAM file ACCTS.DO to OLDACT.DO.

```
10 INPUT "New filespec";FS$
20 NAME "ACCTS.DO" AS FS$
```

renames ACCTS.DO to the input string FS$.

NEW
Erase the Current Program

NEW

NEW erases the current program, sets numeric variables equal to zero, and sets string variables equal to null (""). NEW does **not** change the string space allocation.

Example

```
NEW
```

deletes the current program.

NEXT (See FOR...NEXT)

ON COM GOSUB
Define Communications Interrupt

ON COM GOSUB *line number*

This command defines a communications interrupt subroutine for incoming RS-232C communications. Once BASIC executes ON COM GOSUB, on receiving data over the RS-232C line, it branches to *line number*, regardless of where it currently is in the program. Normally, you'll put this command at the beginning of your program.

(**Note:** You must enable communications interrupt before it can interrupt the program. See **COM ON** for details.)

Example

```
10 ON COM GOSUB 1000
20 COM ON
.
.
1000 OPEN "COM:78N1E" FOR INPUT AS 1
1010 OPEN "IMPDAT.DO" FOR OUTPUT AS 2
1020 LINE INPUT #1,A$
1030 PRINT #2,A$
1040 IF NOT EOF(1) THEN GOTO 1020
1050 CLOSE 1,2
1060 RETURN
```

defines a communications interrupt routine starting at line 1000. When data begins coming in on the RS-232C line, control transfers to line 1000, where it copies the input into a RAM file called ‘‘IMPDAT.DO’’.

ON ERROR GOTO
Define Error Interrupt

ON ERROR GOTO *line number*

ON ERROR defines an error trapping interrupt. After executing this command, if an error occurs elsewhere in the program, BASIC immediately jumps to *line number*. Normally, the routine beginning at *line number* processes the error in some fashion. At the end of the routine, you must either terminate the program (STOP or END) or else return to the program with **RESUME**. See STOP, END, and RESUME for more details.

Example

```
100 ON ERROR GOTO 1000
.
.
200 X = 10000 / Y
.
300 X = 300 / Y
.
.
1000 IF ERR<>11 THEN PRINT
     "Error Code";ERR;"in line ";ERL :
     STOP  ELSE X=100000: RESUME NEXT
```

If an error occurs, BASIC jumps to line 1000. If the error was a division by zero (error #11), then X is set to a high value, 100000, and execution returns to the line following the error line, either line 200 or line 300. If some other error occurred, BASIC prints out the message and stops.

ON KEY GOSUB
Define Function Key Interrupts

ON KEY GOSUB *line number list*

This statement defines interrupts for the Function Keys. After executing this command, pressing the *n*th Function Key tells BASIC to jump to the *n*th *line number* in *line number list*. You may define as many of the Function Keys as you wish — BASIC ignores your pressing of undefined keys.

(**Note:** You must enable the Function Keys before they will interrupt the program. See **KEY ON** for details.)

Examples

```
10 ON KEY GOSUB 1000,2000,3000,,5000
```

defines an interrupt subroutine for Function Key 1, beginning at line 1000, an interrupt subroutine for Function Key 2, beginning at line 2000, an interrupt subroutine for Function Key 3, beginning at line 3000, and an interrupt subroutine for Function Key 5, beginning at line 5000. Function keys 4,6,7, and 8 are left undefined.

ON MDM GOSUB
Define Modem Interrupt

ON MDM GOSUB *line number*

This command defines an interrupt for incoming modem communications. Once BASIC executes ON MDM GOSUB, on receiving data over the modem, it branches to line number, regardless of where it currently is in the program. Normally, you'll put this command at the beginning of your program.

(**Note:** You must enable the modem interrupt before it can interrupt the program. See **MDM ON** for details.)

Example

```
10 ON MDM GOSUB 1000
```

defines a modem interrupt routine beginning at line 1000.

ON TIME$ GOSUB
Define Clock Interrupt

ON TIME$ = *"time"* **GOSUB** *line number*

This command defines an interrupt for a clock condition. *time* is a string expression of the form "HH:MM:SS." When TIME$ equals time, BASIC calls the subroutine at *line number*, regardless of where it currently is in the program. Normally, you'll put this command at the beginning of your program.

(**Note:** You must enable the TIME$ interrupt before it can interrupt the program. See **TIME$ ON** for details.)

Example

```
10 ON TIME$ = "14:20:00" GOSUB 1000
```

defines a clock interrupt for 2:20 PM (14:20:00), beginning at line 1000.

ON...GOTO
Branch on Expression

ON *numeric expression* **GOTO** *line number list*

ON...GOTO evaluates *numeric expression* to an integer *n,* then branches to the *n*th *line number* in the list. *numeric expression* must evaluate to a non-negative number, which, if zero or greater than the number of line numbers in the list, tells BASIC to continue execution without branching.

Example

```
10 ON X GOTO 100,200,300
```

branches to 100, 200 or 300, depending if X equals 1,2, or 3, respectively.

ON...GOSUB
Branch on Expression

ON *numeric expression* **GOSUB** *line number list*

ON...GOSUB evaluates *numeric expression* to an integer *n,* then calls the subroutine beginning at the *n*th *line number* in the *list*. *numeric expression* must evaluate to a non-negative number, which, if zero or greater than the number of line numbers in the list, tells BASIC to continue execution without branching.

Example

```
10 ON X GOSUB 100,200,300
```

calls the subroutine beginning at line 100,200 or 300, depending if X equals 1,2, or 3, respectively.

OPEN
Open a File for I/O

OPEN *"device:filename or configuration"* **FOR** *mode* **AS** *file number*

OPEN allocates a buffer for a file on the given device. *device* may be **RAM, CAS, COM, LCD, LPT,** or **MDM.** *file number* is the buffer number assigned to the file. *mode* can be:

OUTPUT	specifying data will be written sequentially to the file, starting at the beginning of the file
INPUT	specifying data will be read sequentially from the file, starting at the beginning of the file
APPEND	specifying that data will be written sequentially to the file, adding records to the end of the file

RAM Files: *filename* is a string of up to six characters, the first of which is a letter, plus a two character extension which must be .DO. *mode* can be OUTPUT, INPUT, or APPEND.

Cassette Files (CAS): *filename* is a string of up to six characters, the first of which is a letter. *mode* can be OUTPUT or INPUT.

Communications Files (COM): *configuration* consists of a five character string of the form *rwpbs*, where

r	**Baud Rate** This is a number from 1 to 9, where 1=75; 2=110; 3=300; 4=600; 5=1200; 6=2400; 7=4800; 8=9600; 9=19200.
w	**Word Length** This is a number from 6 to 8, where 6=6 bits; 7=7 bits; 8=8 bits.
p	**Parity** Either E,O, or N, where E=even; O=odd; I=Ignore; N=none.
b	**Stop Bits** Either 1 or 2, where 1=1 stop bit; 2=2 stop bits.
s	**XON/XOFF Status** Either E or D, where E=enable; D=disable.

mode can be INPUT or OUTPUT.

Modem Files (MDM): *configuration* consists of a four character string of the pattern *wpbs,* defined above. (BASIC automatically sets the baud rate to 300 baud.)

Screen Files (LCD): *mode* must be OUTPUT. There is no *configuration*.

Printer Files (LPT): *mode* must be OUTPUT. There is no *configuration*.

(**Note:** If your program uses more than one file at once, you must reset MAXFILES.)

Examples

```
10 OPEN "RAM:ACCT.DO" FOR APPEND AS 1
```

opens a RAM file called ACCT.DO for appending, and assigns it the file number "1."

```
10 OPEN "CAS:" FOR OUTPUT AS 3
```

opens an output file on cassette and assigns it to file number "3."

```
10 OPEN "COM:6601E" FOR INPUT AS 4
```

opens a communications file for input as file number 4, using 2400 baud, 6 bit words, odd parity, 1 stop bit, and line enable.

```
10 OPEN "MDM:6E1E" FOR INPUT AS 4
```

opens a modem file for input as file number 4, using 6 bit words, even parity, 1 stop bit, and line enable.

```
10 OPEN "LCD:" FOR OUTPUT AS 1
```

opens a screen file as file number 1.

```
10 OPEN "LPT:" FOR OUTPUT AS 1
```

opens a printer file as file number 1.

OUT
Output a Byte to a CPU Port

OUT *port number, byte value*

This command outputs *byte value* to *port number*. *port number* and *byte value* are numeric expressions in the range 0 to 255.

Example

```
10 OUT 55,100
```

outputs 100 to CPU port 55.

PEEK
Get a Value From Memory

PEEK *(memory address)*

The PEEK function returns the byte value stored at *memory address*. *memory address* and the returned value are both in decimal form.

Example

```
10 A% = PEEK(16999)
```

assigns the byte value at address 16999 to A%.

POKE
Load a Value into Memory

POKE *memory address, byte value*

POKE loads *memory address* with *byte value*. Both must be expressed as decimal numeric expressions.

Example

```
100 POKE 60000, 104
```

loads 104 into address 60000.

POS
Get Screen Position

POS *(dummy numeric expression)*

POS returns the current horizontal Screen position of the Cursor.

Example

```
100 OP% = POS(0)
```

assigns OP% the current horizontal cursor position.

POWER
Automatic Power Down

POWER *numeric expression*

POWER sets the automatic power down period. *numeric expression* has a range of 10 to 255. The Model 100 will automatically turn off after a period of *numeric expression* x 0.1 minutes if you are neither running a program nor entering commands. The default value is 100 (10 minutes).

Example

```
10 POWER 10
```

sets the automatic power down period to one minute (10 X 0.1).

POWER CONT
Prevent Automatic Power Down

POWER CONT

This command disables the automatic power down feature of the Model 100.

Example

```
10 POWER CONT
```

POWER OFF
Turn Off Power

POWER OFF, RESUME

This turns off the power to the Model 100 immediately. RESUME is optional; if present, upon turning the power back on, the Model 100 resumes execution of the program at the statement following the POWER OFF,RESUME. If not present, then the Model 100 returns to the Main Menu upon power up.

Example

```
10 IF TIME$>"11:30:00" THEN POWER OFF
```

turns off the power if the clock is past 11:30 AM.

```
20 POWER OFF,RESUME
30 PRINT "Starting Back Up"
```

turns off the power. When you turn the power back on, BASIC begins execution in line 30.

PRESET
Turn Off an LCD Pixel

PRESET *(x-coordinate, y-coordinate)*

PRESET turns off the LCD pixel at *(x-coordinate, y-coordinate)*. *x-coordinate* may range from 0 to 239, and *y-coordinate* may range from 0 to 63. See also **PSET.**

Example

```
10 PRESET (55,10)
```

turns off the pixel at (55,10).

PRINT
Print Data on the Screen

PRINT *expression list*

This command prints the data in *expression list* onto the Screen, starting at the left-most end of the line. The items in *expression list* are separated by commas or semi-colons. If commas are used, the Cursor automatically advances to the next ''print zone'' before printing the next item. Print zones are at column 0 and column 14. If semi-colons are used, no space is inserted between the items printed on the display.

Positive numbers are printed with a leading blank and all numbers are printed with a trailing blank. Trailing zeroes to the right of the decimal point are not printed out.

No blanks are printed before or after strings. BASIC automatically moves the cursor to the next line after printing the expression list.

You may use a question mark ("?") as an abbreviation for the word PRINT.

Examples

```
100 PRINT "Menu #";I
```

prints MENU # followed immediately by the value of I.

```
200 PRINT I%,J%,K%
```

prints the value of I% starting in column 0, J% in column 15, and K% in column 0 of the next line.

PRINT
Print to a File

PRINT # *file number, expression list*

PRINT # prints or transmits the values of *expression list* to the file opened as *file number*. The items in *expression list* are separated by commas or semi-colons. If commas are used, the Cursor automatically advances to the next "print zone" before printing the next item. Print zones are defined at columns 0, 15, 30, and so on. If semi-colons are used, no space is inserted between the items.

Positive numbers are printed with a leading blank and all numbers are printed with a trailing blank. Trailing zeroes to the right of the decimal point are not printed out. No blanks are printed before or after strings.

You may use a question mark ("?") as an abbreviation for the word PRINT.

Examples

```
100 OPEN "CAS:" FOR OUTPUT AS 1
.
.
200 PRINT #1,A$
```

prints the value of A$ to a file on the cassette tape.

```
100 OPEN "COM:87N1E" FOR OUTPUT AS 4
.
.
200 PRINT #4,10,20,30
```

transmits the values 10, 20, and 30 out the RS-232C lines.

PRINT USING
Formatted Print

PRINT USING *"format"; expression list*

This command prints the data in *expression list* using the specified format. The data items in *expression list* may be separated either by commas or semi-colons. *format* consists of one or more "field specifiers," which describe the type and format of the displayed data. If there are more data items in the list than given formats, BASIC reuses format starting at the left side of the string.

The field specifiers are:

"!"

(String Data) Tells BASIC to print only the first character in the given string.

```
PRINT USING "!";"Tandy"
T
```

"\ *n-spaces* \"

(String Data) Tells BASIC to print 2 + *n* characters from the *string*. If the two 's are typed with no spaces, two characters are printed; with one space, three characters are printed, and so on.

If the string is longer than the field, the extra characters are ignored. If the field is longer than the string, the string is left-justified in the field and padded with spaces on the right.

```
PRINT USING "\   \";"Tandy"
Tand
```

#

(Numeric Data) Specifies one digit position. Digit positions are always filled. If the number to be printed has fewer digits than position specified, the number will be right-justified (preceded by spaces) in the field, with spaces filled in on the left. If the number to be printed is larger than the specified field, BASIC prints out a "%" preceding the number.

```
PRINT USING "#####";5
   5
```

+

(Numeric Data) Inserts the algebraic sign of the number, either at the beginning or end of the number, depending on its occurrence in the format string

```
PRINT USING "+#####";-13
  -13

PRINT USING "#####-";14
  14
```

-

(Numeric Data) For negative numbers, inserts a minus sign either at the beginning or end of the number, depending on its occurrence in the format string. If the number is positive, then BASIC inserts a blank.

```
PRINT USING "-######";14
   14

PRINT USING "#####.##-";0.45
  0.45
```

**

(Numeric Data) Changes any leading blanks to leading asterisks blanks. The ** also counts as two digit positions and must occur on the left side of the format string.

```
PRINT USING "**######";145
****145
```

$$

(Numeric Data) Prints a dollar sign to the immediate left of the formatted number. The **$$** counts as two digit positions, one of which is the dollar sign and must be the first characters of the format string. You may not use the exponential format unless you specify a trailing minus sign.

```
PRINT USING "$$######";450
   $450
```

**$

(Numeric Data) Fills leading spaces to asterisks except for the space to the immediate left of the number, where it inserts a dollar sign. ****$** counts as three digit positions, one of which is the dollar sign.

```
PRINT USING "**$###";12
***$12
```

.

(Numeric Data) Inserts a decimal point. This specifier must be used as part of "#" field string. If the format string specifies that a digit is to precede the decimal point, the digit will always be printed (as 0 if necessary). Digits to the right of the decimal point are rounded as necessary.

```
PRINT USING "#####.##";14.5
   14.50

PRINT USING "#####.##";0.588
    0.59
```

,

(Numeric Data) Inserts a comma every three digits to the left of the decimal point. If the digit to the left of a potential comma is blank, then BASIC inserts a blank instead of the comma. The , must lie between numeric field specifiers (#,$ or **), to the left of the decimal point and counts as a digit position.

```
PRINT USING "#########,";14432
    14,432
```

^^^^

(Numeric Data) Specifies exponential format. The four carats count as four characters in the field and come after the numeric descriptors. Any decimal point position may be specified — the significant digits are left-justified, and the exponent is adjusted.

Unless a leading + or trailing + or − is specified, one digit position will be used to the left of the decimal point to print a space or a minus sign.

```
PRINT USING "***###^^^^";150000
E-0.4
```

(**Note:** The caret (^) is entered by pressing SHIFT 6, and the backslash (\) is entered by pressing GRPH −.)

Examples

```
    10 PRINT USING "\            \ #####,.##  #####,.##";
       A$,IBAL,OBAL
```

If A$ contains the string "Cramer,W.D", IBAL equals 1440.44, and OBAL equals 980.00, then this statement prints:

```
Cramer,W.D      1,440.44      980.00
```

```
200 PRINT USING "$$####.##  ";A,B,C
```

If A contains 34, B contains 44.323, and C contains 12333.33, then this statement prints out:

```
$34.00     $44.32  $12333.33
```

Note that the blanks in the format string **are** significant.

In addition, characters other than the field specifiers are inserted as is, providing there is enough room in the field that BASIC doesn't try to use the characters for conversion. For example,

```
PRINT USING "$#######,.##";4534.34
```

prints:

```
$  4,534.34
```

PRINT # USING
Formatted Print to a File

PRINT *#file number,* **USING** *"format"; expression list*

Formats the data in expression list and sends it to the device opened as file number. See **PRINT #** and **PRINT USING** for more information and examples.

PSET
Turn On LCD Pixels

PSET *(x-coordinate,y-coordinate)*

PSET turns on the LCD pixel at *x-coordinate,y-coordinate,* where *x-coordinate* is a numeric expression ranging from 0 to 239 and *y-coordinate* is a numeric expression ranging from 0 to 63.

Example

```
10 PSET (40,45)
```

turns on the pixel at 40,45.

READ
Read Values From a DATA List

READ *variable list*

This command reads an appropriate number of values from a DATA statement and stores the values in the variables of *variable list*. The values in the DATA statements must match in type (string or numeric) with the variables in variable list.

The first time BASIC executes a READ command, the first value in the first DATA statement is used, the second time, the second value in the DATA statement is read, and so on. When all the items in the first DATA statement have been read, the next READ uses the first value in the second DATA statement, and so on.

To reuse the values of the DATA command, use the RESTORE command.

See also **DATA** and **RESTORE.**

Example

```
100 DATA 0.4, 0.2, "Trinity River"
120 READ A,B%,C$
```

assigns A the value 0.4, B% the value 0.2, and C$ the string Trinity River.

REM
Comment

REM *comment statement*

REM signifies to BASIC that the remainder of the line is a comment. Since BASIC ignores everything following REM, *comment statement* must be the last statement of the line.

You may abbreviate REM with an apostrophe. If the comment follows another BASIC command, then you must either use the ’ or else precede REM with a colon.

Examples

```
10 REM This program finds the standard deviation

10 ' This program finds the standard deviation

100 AVE = SUM / TT   'Calculate the average

100 AVE = SUM / TT   :REM Calculate the average
```

RESTORE
Reset the DATA Statement Pointer

RESTORE *line number*

This command resets the DATA statement pointer to the first item in the DATA statement on *line number* so that a READ command can access the same values more than once. *line number* is optional; if omitted, BASIC uses the first DATA statement.

See also **DATA** and **READ.**

Example

```
100 DATA "Nuts","Bolts","Screws","Hammers"
.
.
300 READ ITEM$(1),ITEM$(2),ITEM$(3),ITEM$(4)
.
.
600 RESTORE 100
610 READ CT$(1),CT$(2),CT$(3),CT$(4)
```

Line 300 assigns the strings of the DATA statement in line 100 to ITEM$'s 1 through 4. Line 600 resets the DATA pointer so that line 610 reassigns the strings to CT$'s 1 through 4.

RESUME
Resume Execution After an Error

RESUME *line number*

RESUME ends an error handling routine by branching to *line number* where BASIC begins normal execution. If *line number* is null or 0, then BASIC returns to the line which caused the error. You may also specify NEXT in which case BASIC returns to the line immediately following the error causing line.

Example

```
1000 IF ERR = 18 THEN PRINT @0, "Printer
Not Ready!!!": RESUME
1010 .
.
```

If an I/O error occurs, then BASIC prints the message and resumes execution at the offending line. Otherwise, BASIC proceeds to line 1010.

RIGHT$
Return Right Portion of a String

RIGHT$ *(string expression,count)*

RIGHT$ returns the right-most *count* characters of *string expression*. *count* is a numeric expression.

Example

```
10 SEC$ = RIGHT$(TIME$,2)
```

assigns the current second count to SEC$.

RND
Return Pseudo-Random Number

RND *(numeric expression)*

RND returns a pseudo-random number between 0 and 1. If *numeric expression* is non-zero, then RND returns a new random number. If *numeric expression* equals 0, then RND returns the last random number generated.

Example

```
20 PRINT RND(1)
30 PRINT RND(0)
```

Prints the same random number twice.

(**Note:** RND always generates the same random number series. If your application requires a different random number starting the sequence each time, you can use the clock to establish a starting point in the sequence. For example, the following routine points the random number generator to one of 60 starting points in the generator:

```
10 SEC = VAL(RIGHT$(TIME$,2))
20 FOR I=1 TO SEC
30 DUMMY = RND(1)
40 NEXT I
```

RUN
Execute a New BASIC Program

RUN *"device:filename or configuration"*,**R**

RUN loads and runs a BASIC program from *device*. *filename* consists of a string of one to six characters, the first of which is a letter. *device* may be **RAM, CAS, COM** or **MDM** if *device* is **RAM**, then you may include the optional extension **.BA** or **.DO**. If *device* is **CAS**, then you use no extension.

For **COM**, configuration consists of a five character string of the pattern *rwpbs,* where

r	**Baud Rate** This is a number from 1 to 9, where 1 = 75; 2 = 110; 3 = 300; 4 = 600; 5 = 1200; 6 = 2400; 7 = 4800; 8 = 9600; 9 = 19200.
w	**Word Length** This is a number from 6 to 8, where 6 = 6 bits; 7 = 7 bits; 8 = 8 bits.
p	**Parity** Either E,O,I, or N, where E = Even; O = Odd; I = Ignore; N = None.
b	**Stop Bits** Either 1 or 2, where 1 = 1 stop bit; 2 = 2 stop bits.
s	**XON/XOFF Status** Either E or D, where E = Enable; D = Disable.

For **MDM,** *configuration* consists of a four character string of the pattern *wpbs,* defined above. (BASIC automatically sets the baud rate to 300 baud.)

R is optional; if present, it tells BASIC keep all open files opened. If omitted, BASIC closes any currently opened files before running the new program.

Examples

```
1000 RUN "PART2.BA",R
```

loads and executes the RAM file PART2.BA, keeping all open files open.

```
100 RUN "MDM:7E2E"
```

loads and executes the BASIC program coming in over the modem lines.

RUN
Execute the Current BASIC Program

RUN *line number,* **R**

Run clears all variables and begins execution of the current program, starting at *line number*. *line number* is optional; if omitted, BASIC starts execution at the first line of the program. **R**, if present, tells BASIC to leave currently opened files open. If not present, BASIC closes all files before executing the program.

Examples

```
RUN 100
```

Clears all variable values and starts executing the program at line 100.

```
RUN,R
```

clears all numeric and string variables and begins execution of the current program. Open files are left open.

RUNM
Load and Execute a Machine-Language Program

RUNM "CAS or RAM:*filename***"**

Loads and executes the machine-language program stored as *filename*. *filename* consists of a string of up to six characters, the first of which is a letter. For **RAM** files, you may optionally include the extension **.CO**. The word **RAM** is optional; if no device is specified, BASIC assumes RAM.

Cassette files require no extension. *filename* is optional; if omitted, BASIC loads and runs the first machine-language program it finds.

BASIC closes all open files before running the machine-language program.

(**Note:** For RAM files, the program must be one which is executable from the Main Menu — not a BASIC subroutine!)

Examples

```
RUNM "MEMTST"
```

loads the program MEMTST.CO from RAM and then executes it.

```
RUNM "CAS:"
```

loads and runs the first machine-language program found on the cassette tape.

SAVE
Save a BASIC Program

SAVE *"device:filename* or *configuration"***,A**

SAVE writes the current BASIC program to the specified device. *device* may be **RAM, CAS, COM,** or **MDM**. *filename* consists of a string of 1 to 6 characters, the first of which is a letter. **RAM** filenames are optionally followed by the extension **.BA** or **.DO**. (if not present, BASIC adds an extension automatically).

The word **RAM** is also optional; if no device is named, BASIC assumes RAM. If filename already exists in RAM, BASIC writes over the old file. If device is **CAS**, there is no extension.

If device is **COM,** *configuration* consists of a five character string of the pattern *rwpbs,* where

r	**Baud Rate** This is a number from 1 to 9, where 1=75; 2=110; 3=300; 4=600; 5=1200; 6=2400; 7=4800; 8=9600; 9=19200.
w	**Word Length** This is a number from 6 to 8, where 6=6 bits; 7=7 bits; 8=8 bits.
p	**Parity** Either E,O,I, or N, where E=Even; O=Odd; I=Ignore; N=None.
b	**Stop Bits** Either 1 or 2, where 1=1 stop bit; 2=2 stop bits.
s	**XON/XOFF Status** Either E or D, where E=Enable; D=Disable.

If device is **MDM**, *configuration* consists of a four character string of the pattern *wpbs,* defined above. (BASIC automatically sets the baud rate to 300 baud.)

BASIC requires no configuration or filename for **LPT** or **LCD** files.

A is optional; if present, BASIC saves the file in ASCII format. If not present, BASIC saves the file in a compressed format.

(**Note:** You must save BASIC files in ASCII format if you intend to merge them.) **COM, MDM, LCD,** and **LPT** all write the current program to their corresponding device in ASCII format.

Examples

```
SAVE "TIMSET"
```

writes the current BASIC program to the RAM file TIMSET.BA.

```
SAVE "PART3",A
```

writes the current BASIC program to the RAM file PART3.DO. The file is stored in ASCII format.

```
SAVE "CAS:CLOCK"
```

writes the current program to cassette tape naming the file CLOCK (identical to the command CSAVE "CLOCK").

```
SAVE "MDM:7N1E"
```

sends the current program out the modem, using the configuration 7 bit words, no parity check, 1 stop bit, and line enable.

```
SAVE "COM:58E2E"
```

sends the current program out the RS-232C line using the configuration of 1200 baud, 8 bit words, even parity, 2 stop bits, and line enable.

```
SAVE "LPT:
```

writes the current program on the printer (identical to LLIST).

```
SAVE "LCD:"
```

writes the current program to the Screen (identical to LIST).

SAVEM
Store a Machine-Language Program

SAVEM "CAS or **RAM:***filename, start address, end address, entry address*

SAVEM writes the program stored from *start address* to *end address* onto cassette tape or RAM, under the name *filename*. *entry address* is optional; if not present, BASIC assumes the program *entry address* is the same as the *start address*.

filename consists of a string of one to six characters, the first of which is a letter. For **RAM** files, you may include the extension **.CO**. Cassette files require no extension. If you don't specify a device, BASIC assumes **RAM**.

(**Note:** SAVEM "CAS: functions identically to the command CSAVEM.)

Examples

```
SAVEM "CAS:MEMTST",50000,50305,50020
```

writes the program stored from addresses 50000 to 50305 with the entry point at 50020 onto cassette tape, giving the file the name ''MEMTST.''

```
SAVEM "MEMTST",50000,50305,50020
```

writes the program stored from addresses 50000 to 50305 with the entry point at 50020 into cassette tape, giving the file the name ''MEMTST.CO''.

SCREEN
Locks/Unlocks LABEL Line

SCREEN *on/off*

SCREEN locks or unlocks the bottom (LABEL) line on the Display for scrolling. *on* is 0,0 and *off* is 0,1.

Example

```
SCREEN 0,0
```

causes the LABEL line to disappear and allows you to scroll with all eight lines.

```
SCREEN 0,1
```

causes the LABEL line to reappear.

SGN
Algebraic Sign

SGN*(numeric expression)*

This expression returns a −1 for negative numbers, 0 for zero, and 1 for positive numbers.

Example

```
200 TTL = 10 * SGN(CR)
```

sets TTL equal to either 10, 0, or −10, depending on whether CR is positive, zero, or negative.

SIN
Trigonometric Sine

SIN *(numeric expression)*

SIN returns (in radians) the trigonometric sine of *numeric expression.*

Example

```
100 Y = SIN(1.5)
```

assigns Y the value 0.99749498660406.

SOUND
Output a Tone

SOUND *pitch, length*

SOUND "plays" a given pitch for the given length. *length* ranges from 0 to 255. Dividing length by 50 gives the approximate length in seconds. *pitch* ranges from 0 to 16383, with the larger values corresponding to higher pitches. The values of *pitch* corresponding to musical notes are shown below.

	Octave				
Note	1	2	3	4	5
G	12538	6269	3134	1567	783
G#	11836	5918	2959	1479	739
A	11172	5586	2793	1396	698
A#	10544	5272	2636	1318	659
B	9952	4976	2484	1244	622
C	9394	4697	2348	1174	587
C#	8866	4433	2216	1108	554
D	8368	4184	2092	1046	523
D#	7900	3950	1975	987	493
E	7456	3728	1864	932	466
F	7032	3516	1758	879	439
F#	6642	3321	1660	830	415

SOUND ON/OFF
Enable/Disable Sound

SOUND ON or **OFF**

SOUND ON tells BASIC to "beep" when:

1) You're loading from cassette.
2) The Model 100 is waiting on a carrier signal from the telephone modem lines.

SOUND OFF disables the "beep" under these circumstances. The cold start default is SOUND ON.

(**Note:** SOUND ON and SOUND OFF do not effect any of the other sound generating commands, such as BEEP and SOUND.)

SPACE$
String of Spaces

SPACE$*(length)*

This function returns a string of *length* spaces.

Example

```
100 B$ = SPACE$(20) + A$
```

sets B$ equal to a string of 20 spaces followed by the string stored in A$.

SQR
Square Root

SQR*(numeric expression)*

SQR returns the square root of *numeric expression*. *numeric expression* must be a positive number.

Example

```
10 C = SQR(A^2 + B^2)
```

sets C equal to the square root of the sum of A^2 and B^2.

STOP
Stop Execution

STOP

STOP stops execution of a BASIC program at some point other than the physical end. STOP is primarily a "debugging" aid. By inserting STOP commands inside your program, you can examine or change the values of variables, and then resume execution of the program (with the **CONT** command) at the point following the STOP command.

Example

```
100 FOR I=1 TO 100
110 B$(I) = MN$ + DESC$(I) + MID$(TIME$,1,2)
111 STOP
120 NEXT I
```

stops execution of the program at line 111. Typing CONT will begin execution at line 120 (providing you have not altered the BASIC program).

STR$
Convert a Number to a String

STR$(*numeric expression*)

STR$ converts *numeric expression* to its string represention. This function is the inverse of **VAL.**

Example

```
B$ = "$" + STR$(BAL) + ".00"
```

If BAL contains the value 133, then this statement sets B$ equal to $133.00.

STRING$
Define a String of Characters

STRING$ *(length, character)*

STRING$ returns a string of the given *length* composed of *character*. *length* may range from 0 to 255. *character* is either a string expression or numeric expression; if it is a string expression, only the first character of the string is duplicated. If it is a numeric expression, it must evaluate to a number between 0 and 255.

Example

```
PRINT STRING$(20,"*")
```

prints a string of 20 asterisks.

```
PRINT STRING$(40,239)
```

prints a string of 40 solid blocks (239 is the ASCII code for a solid block).

TAB
Skip to Specified Position

TAB *(numeric expression)*

TAB skips *numeric expression* spaces before printing the next data item. *numeric expression* ranges between 0 and 255.

You may only use this function as part of the data list of an output statement. Note that using commas in the data list may produce undesirable results.

Examples

```
10 PRINT TAB(30);"Table 1"
```

prints "Table 1" starting in column 30.

```
20 LPRINT TAB(10);"Total";TAB(20);"Number";
   TAB(30);"Balance"
```

skips 10 spaces and prints Total on the printer, skips another 10 spaces and prints Number, and finally skips another 10 spaces and prints Balance.

TAN
Trigonometric Tangent

TAN *(numeric expression)*

TAN returns the tangent of *numeric expression. numeric expression* must be in radians.

Example

```
10 SLOPE = TAN(THETA)
```

assigns SLOPE the value of the tangent of THETA.

TIME$
Current Time

TIME$ keeps track of the current time, in the form of a string variable. You may access it like any string variable, including resetting the time. The time string has the form "HH:MM:SS", where $00 \leq HH \leq 23$, $00 \leq MM \leq 59$, and $00 \leq SS \leq 59$. BASIC automatically updates TIME$, including changing from 23:59:59 to 00:00:00.

(**Note:** BASIC allows values up to 29 for HH. However, such values have no meaning and prevent TIME from ever returning to 00:00:00.)

Examples

```
PRINT TIME$
```

prints the current time.

```
TIME$="10:00:00"
```

sets the time to 10:00 AM.

TIME$ ON/OFF/STOP
Enable/Disable Time Interrupt

TIME ON/OFF/STOP

TIME ON enables ON TIME$ interrupting and TIME OFF disables ON TIME$ interrupting. TIME STOP disables the interrupt, however, BASIC "remembers" that the ON TIME$ condition occurred, so that if you issue a TIME$ ON command, BASIC jumps immediately to the interrupt subroutine.

See also ON TIME$.

Example

```
10 ON TIME$="20:00:00" GOSUB 1000
20 TIME$ ON
.
.
1000 TIME$ = "19:00:00"
1010 TIME$ OFF
1020 RETURN
```

The first time that the clock reaches 20:00:00, BASIC jumps to line 1000, resets the clock, and returns to what it was doing before the subroutine call. The next time the clock reaches 20:00:00, nothing happens because the interrupt was disabled in line 1010.

VAL
Convert Strings To Numbers

VAL *(string expression)*

VAL converts *string expression* to a numeric representation of the string. If *string expression* contains non-numeric characters, VAL returns only the value of the leading number, if any. **VAL** is the inverse of the function STR$.

Examples

```
5 B$ = "100.44824"
10 A = VAL(B$)
```

sets A equal to 100.44824.

```
5 B$ = "no balance"
10 A = VAL(B$)
```

sets A equal to 0.

```
5 B$ = "3.00313354E33"
10 A = VAL(B$)
```

sets A equal to 3.00313354 X 10^{33}.

VARPTR
Get Address of a Variable

VARPTR *(variable name)*

VARPTR returns the memory address of *variable name*. If you haven't yet used *variable name,* then VARPTR causes an error condition. This function may be useful in conjunction with PEEK and POKE, as well as CALL. Note that the returning address is an integer value, expressed in decimal form, hence memory addresses over 32767 return negative values.

Example

```
LINK(I) = VARPTR(B$)
```

sets LINK(I) equal to the memory address of B$.

MODEL 100

PART IV/ APPENDICES

Appendix A / Connecting the Model 100 to Optional Equipment

Before connecting any optional equipment to the Model 100, be sure that the Computer and the optional device are both turned **OFF**.

Once connections are made, avoid setting the Computer on top of any connecting cables.

Optional Equipment Power ON/OFF Sequence

When powering up a Model 100 system, turn the Computer's power ON, then the optional device. To turn the power OFF, always turn the optional device OFF first, then the Computer.

Cassette Recorder

For best results, we recommend that you use the Radio Shack ***CCR-81 Computer Recorder*** (26-1208) which includes a Recorder-to-Computer cable (26-1207).

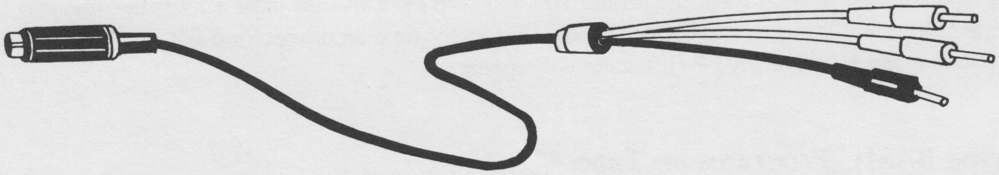


Figure A-1. CCR-81 Recorder-to-Computer Connection Cable

1. Connect the large round plug of the Connection Cable to the connector labeled **CASSETTE** on the back side of the Model 100.
2. Connect the black plug into the Recorder's **EAR** connector.
3. Connect the larger gray plug into the Recorder's **AUX** connector.
4. Connect the smaller gray plug into the Recorder's **REM** connector.

Saving Text Files

Once a Recorder is properly connected to a Model 100, follow these steps to save a Text file on tape:

1. Press the Recorder's **RECORD** and **PLAY** buttons together until they lock. You do not need to set the volume; the Recorder does this for you automatically during recording operations.
2. From the Model 100 Main Menu, open the file you wish to save by placing the Cursor over the file name and pressing **ENTER**.

3. Once the beginning of the file is displayed on the Screen, press the **SAVE** Function Key (**F3**).
4. The prompt `Save "` will appear. Type a file name (no longer than six characters) which you want to assign to the file you're saving.
5. Press **ENTER**.

The Recorder will start turning automatically. It will stop after the file has been saved. Note: Motor control must be used to start and stop the recorder.

Loading Text Files

1. Rewind the tape, press the **PLAY** button until it locks and set volume between 4 and 6.
2. Open a file or create a new file (see **"Creating Text Files"** in Chapter 8).
3. Press the **LOAD** Function Key (**F2**). The prompt `Load "` will appear.
4. Type the file name assigned to the file when it was SAVEd.
5. Press **ENTER**.

The Model 100 will produce a high-pitched tone, indicating it is searching for the file. Once the file is located, the prompt `Found:`*filename* (where *filename* is the name you specified) will appear.

If the tape contains several files, the Model 100 will skip over all files until it finds the one you specified. You will know this is happening because every time an unspecified file is encountered, the message `SKIP:`*filename* will appear.

Storing BASIC Programs on Tape

BASIC programs (i.e., any file with the extension **.BA**) must be stored from the BASIC Application Program.

The reason for doing this is that a program stored as a .BA file will be executed immediately when you press **ENTER** after positioning the Cursor on top of it. However, to store the program from a .BA file you would first have to break program execution and then store it.

Load a program file into BASIC:

1. Access BASIC from the Main Menu.
2. Press **LOAD** (**F2**) and type the name of the **.BA** file you wish to store on tape.
3. Press **ENTER**.

When the program has been loaded into BASIC, you may begin storing it on tape:

1. Press the **RECORD** and **PLAY** buttons together until they lock. You do not need to set the volume; the cassette recorder does this for you automatically during recording operations.

2. Either type `CSAVE` *''filename''* or press the **SAVE** Function Key ((F3)), wait for the prompt `Save to:`, and type:

   `CAS:`*filename*

   (where *filename* is the name under which the program will be stored). The file name cannot exceed six characters in length.

4. Press (ENTER).

The Recorder's motor will start automatically. It will then stop after the file has been stored.

Notice that when saving a program on tape, using the general device command **SAVE** it is necessary to specify the peripheral device by typing CAS: before assigning a name to the program. If you fail to do so, an FC (Illegal Function Call) error will occur.

Loading a BASIC Program from Tape

1. Rewind the tape, press the **PLAY** button until it locks, and set volume between 4 and 6.
2. Access the BASIC Application Program from the Main Menu.
3. Either type `CLOAD` *''filename''* or press the **LOAD** Function Key ((F2)), wait for the prompt `Load from "`, and type:

   `CAS:`*filename*

   (where *filename* is the name which was assigned to the program when it was SAVEd).

4. Press (ENTER).

The Model 100 will produce a high-pitched tone to indicate it is searching for the program. Once the program is located, the prompt at the bottom of the Screen will change to `Found:`*filename*, where *filename* is the name under which the program was SAVEd.

If the tape contains several programs, the Model 100 will skip over them until it finds the one you specified. You will know this is happening because every time an undesired program is found, the message `SKIP:`*filename* appears at the bottom of the Screen.

Connecting the Model 100 to a Printer

The Model 100 can provide you with ''hard copies'' of programs, documents, listings, or reports if you connect it to any Radio Shack Parallel Printer such as the Daisy Wheel II, the DWP-410, the DMP-400, or the DMP-500. To connect the Model 100 to any of these printers, you'll need the optional ***Model 100 Printer Cable*** (26-1409).

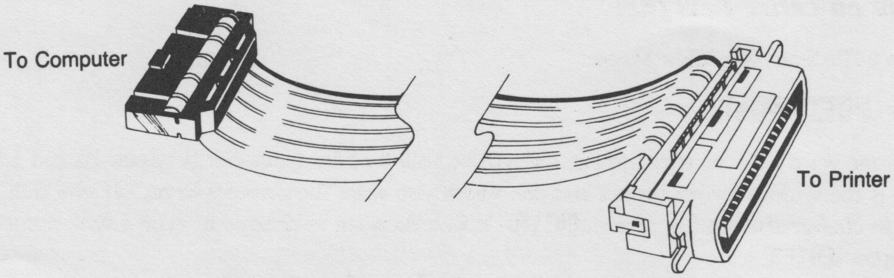


Figure A-2. Model 100 Parallel Printer Cable

1. Connect the Cable's large connector to the Printer.
2. Connect the cable's small connector to the **PRINTER** Connector on the rear panel of the Model 100.

There is only one way to attach the Cable Connector to the printer and Computer. If it is difficult to attach the cable to the Computer or printer, the cable connector may be upside-down. Turn the connector the other way and try connecting it again. Do not force the cable!

The cable attached to the small connector must exit towards the bottom of the Computer (see Figure A-3).

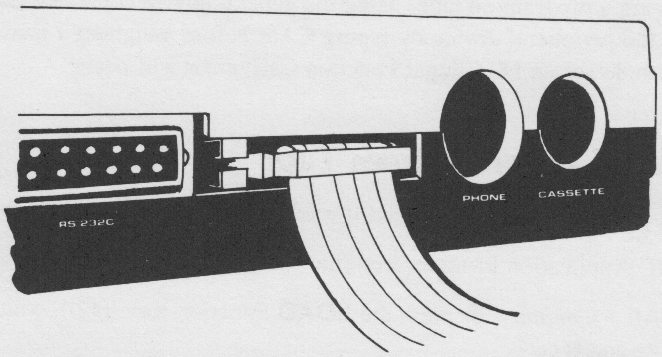


Figure A-3. Printer Cable attached to the Model 100

See your printer's operation manual for details on using the printer (loading paper, inserting ribbons, setting parameters, etc.).

Using the Model 100 with a Printer

Once a printer has been connected to the Model 100, you have several printing options:

- Print whatever appears on the Display
- Print an entire file
- Print program listings
- Print the retrieved information from the ADRS.DO or NOTE.DO files directly to the printer.

To print whatever appears on the Display:

1. Press the (PRINT) Command Key.

To print an entire Text file:

1. Open a file from the Main Menu.
2. Press (SHIFT) (PRINT).
3. When the `Width:` prompt appears, it will be followed by a number between 10 and 132. (This is the width of your Printer and the width you want the printout to be.) If you don't wish to change this setting, press (ENTER). If you do want to change it, type a new number and press (ENTER).

If you want to print the file as is, just press (ENTER).

To print a program listing:

1. Access BASIC from the Main Menu.
2. Load the program file into BASIC.
3. Type `LLIST` **ENTER**.

The entire program will be listed on the printer.

If you are only interested in certain line numbers, type the first and last line number you want listed. For example:

```
LLIST 60-125
```

See the **LLIST** command in Part III of this manual for more details.

To print program results:

You will have to include the LPRINT command as part of your program if you want calculation results displayed on a printer.

See the **LPRINT** command in Part III of this manual for more details.

To retrieve information from the ADRS.DO or NOTE.DO files directly to the printer:

1. Access the ADDRSS or the SCHEDL Application Programs.
2. Press the **LFND** Function Key (**F5**) and type the characters used to identify the information you want to get.
3. Press **ENTER**.

Pressing **BREAK** during any of the above printing operations will cause the printer to stop printing.

Connecting the Model 100 to Other TRS-80 Computers

The Model 100 can be connected to other TRS-80 Computers (Model III, Model II, and Model 16, for instance) via the RS-232C Interface on both computers.

Use a standard **DB-25 RS-232C Cable** (such as the five foot **RS-232C Cable,** 26-4403) to make the connection. Radio Shack also provides RS-232C cables up to 100′ in length.

When connecting the Model 100 to another TRS-80, it is necessary to use a ***Null Modem Adapter*** (26-1496).

However, the Null Modem Adapter will not fit securely on the RS-232C Connector of the Model 100. For this reason, attach the Null Modem to the RS-232C connector of the ‘‘other’’ TRS-80 before linking the Computers with an RS-232C Cable.

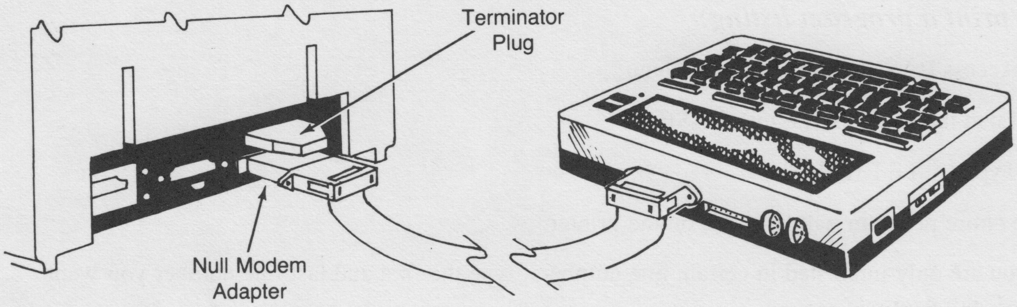

Terminator
Plug
Null Modem
Adapter

Figure A-4. Model 100 to Model II/16 Connection

To connect the Model 100 to the TRS-80 Model III, you will have to use an RS-232C cable and a Cable Extender with a Null Modem Adapter in between. See Figure A-5.

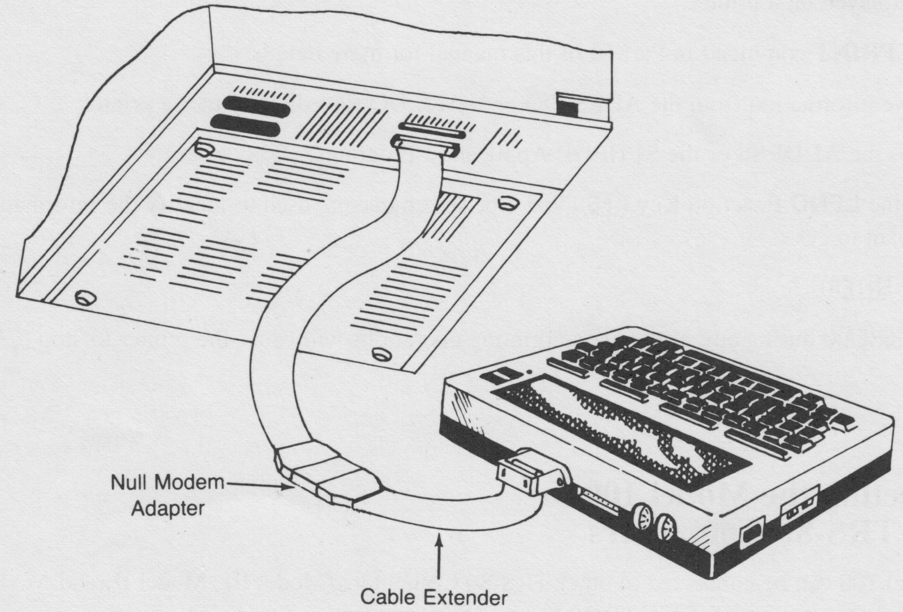

Null Modem
Adapter
Cable Extender

Figure A-5. Model 100 to Model III Connection

Appendix B / Notes on Power On/Off

Warm vs. Cold Power On/Off

There are two ways to start the Model 100 — Warm and Cold Starts.

Simply turning on the Power Switch constitutes a Warm Start. In a Warm Start, any programs, data, or files that you have previously created remain effectively protected and appear on the Main Menu under the names you have assigned them.

Conversely, a Cold Start clears all the memory contents, including any information you may have entered into the Model 100, even the time and date. If you ever increase the memory capacity of the Computer or install another ROM, it will be necessary to perform a Cold Start. It is advisable, therefore, to save important programs or data on tape before you perform a Cold Start.

A Cold Start can be done by pressing **CTRL** **PAUSE** while turning on the Power Switch or while pressing and holding RESET.

Changing the Auto-Power Off Settings

When you use the Model 100 initially, the automatic Power-Off limit is set to 10 minutes. However, you can change this time limit to a greater or lesser value.

Automatic power-off can occur at any time between 0 and 25 1/2 minutes. You can also have the Model 100 disregard the automatic power-off function completely. If you do this and forget to set the Power Switch to OFF, however, the batteries can be exhausted within a few hours.

Changing and overriding the Automatic Power-Off are done using the BASIC Application Program.

To change the Automatic Power-Off time limit:

1. Access the BASIC Application Program from the Main Menu.
2. Type: **POWER** *n* **ENTER** where *n* can be any number from 10 to 255. The Model 100 multiplies the number you input times 6 seconds.

For instance, typing `POWER 10` **ENTER** would set the automatic power-off at 1 minute (10 × 6 seconds = 60 seconds).

To override the Automatic Power-Off time limit:

1. Access the BASIC Application Program from the Main Menu.
2. Type: `POWER CONT` **ENTER**.

Appendix C / Sample Sessions

Transfering Data Between the Model 100 and the Model II

You may transfer data between the Model 100 and the Model II using the Terminal program on the Model II and the TELCOM program on the Model 100. Model 100 programs must be stored in ASCII format (files with a .DO extension — either Text files or else BASIC program files SAVEd with the **A** option.)

First, make the preliminary hardware and software connections by following these steps:

1. Connect the two computers via an ***RS-232C Cable*** (26-4403) and a ***Null Modem Adapter*** (26-1496). See Figure A-4 in Appendix A.
2. Insert a TRSDOS System Diskette which contains the TERMINAL Program into the Model II. (All standard TRSDOS 2.0 System Diskettes contain the program).
3. Initialize the drive by typing:

   `I` **ENTER**

   and then initialize the RS-232C port by typing:

   `SETCOM A=(4800,8,E,1)` **ENTER**

   You may use other SETCOM parameters for the Model II as long as the Model 100 can match them. The settings used in this example allow the fastest data transfer.
4. Now enter the Model II TERMINAL program by typing:

   `TERMINAL` **ENTER**
5. On the Model 100, move the Menu Cursor to the word `TELCOM` and press **ENTER**. The TELCOM prompt will appear. Press **STAT** (**F3**) and type:

   `78E1E` **ENTER**.

   This sets the Model 100 communication parameters to match the parameters you previously set on the Model II.
6. Enter the Model 100 Terminal Mode by pressing **TERM** (**F4**).

Now that the Computers are connected and have matching communication parameters, you may transmit files to and from the two machines.

Downloading File from the Model II to the Model 100

To download a Model II file to the Model 100, follow these steps:

1. On the Model 100, press **DOWN** (F2). TELCOM will prompt you for a file name. Enter the name of the new file, using the **.DO** extension (TELCOM will add this extension if you don't supply it), then press ENTER. The label `Down` will appear in reverse video.
2. On the Model II, type `G` ENTER (for get disk file), and enter the filespec of the file to be transferred to the Model 100.
3. Once the file is in RAM, type `X` ENTER on the Model II to transmit the file. As the file goes to the Model 100, TELCOM displays the information being transfered on the Model 100 Screen.
4. When all of the data has been transmitted, press F8 on the Model 100 to exit the Terminal Mode. When prompted `Disconnect?`, press Y ENTER.
5. Press BREAK on the Model II to return to the TERMINAL Menu. Then type `Q` ENTER to exit the TERMINAL program.

The Model II file is now stored in the Model 100's memory under the file name you specified.

Uploading Model 100 Text Files to a Model II

To transmit a Model 100 file to a Model II, follow these steps:

1. Enter the TERMINAL Program on the Model II by typing `T` ENTER.
2. On the Model 100, press **UPLOAD** (F3).
3. TELCOM prompts you for the name of the file to be uploaded. Type the file name (remember that it must have the .DO extension!) and press ENTER.
4. TELCOM will then prompt you for width. This refers to the maximum number of characters transmitted before transmitting a carriage return. Enter an appropriate value, then press ENTER. Remember that the Display on the Model 100 is 40 characters wide.

   If you want to send the file ''as is,'' simply press ENTER.
5. As the file is transferred, the Up label appears in reverse video, and the transmitted data appears on the Model II Screen.
6. When the transmission is complete, the Up label returns to normal video.

   On the Model II, press BREAK to return to the TERMINAL menu. Now press `C` ENTER to save the RAM buffer to a disk file. TERMINAL prompts you for the file name. Enter any valid file name. To exit from the TERMINAL program on the Model II, type `Q` ENTER.
7. To exit the TELCOM program on the Model 100, press F8 to exit the Terminal Mode. When prompted `Disconnect?`, press Y ENTER. Now press F8 again to return to the Main Menu.

The Model 100 file is now stored on your Model II disk, under the specified file name. The new file will have a Record Length of 1.

Sample Session #2: Using an Auto Log-on Sequence

The following BASIC program illustrates how a BASIC program can be used to perform the Model 100 TELCOM's Automatic Log-on Sequence when dialing the Dow Jones Information News Service.

After accessing Dow Jones, the program requests ticker symbols and news information for a specified company. This information is then stored in a file named QUOTE.DO.

To examine these quotes, simply position the Cursor (at the Main Menu) over the word `QUOTE.DO` and press **ENTER**.

Enter BASIC and type in the program exactly as it is listed below except substitute your local access phone number and secret password in the Automatic Log-on Sequence in line 20. Also, in lines 5000 and 5010, substitute the Dow Jones codes you would like to use.

For more details on Dow Jones codes and log-on procedures, see your **Dow Jones Information Service User's Guide.**

For details on Model 100 BASIC, see Part III of this manual.

```
5 MAXFILES=3
10 ST$=CHR$(19)
20 PH$="5551234<?pA?PDOW1;;?WDJNS^M?Ppassword^M>"
30 M=VARPTR(PH$)
40 AD=PEEK(M+1)+(PEEK(M+2)*256)
50 CALL 21200
60 CALL 21293,0,AD
70 CLS
80 OPEN "MDM:7I1D" FOR INPUT AS 1
90 OPEN "MDM:7I1D" FOR OUTPUT AS 2
100 OPEN "QUOTE.DO" FOR APPEND AS 3
110 Z$=INPUT$(1,1)
120 IFZ$<>ST$THEN 110
130 PRINT #3,DATE$;" ";TIME$
140 PRINT "STARTING QUOTES REQUEST"
150 READ N
160 FOR I=1 TO N
170 READ Q$
180 PRINT #2,Q$
190 GOSUB 4000
200 PRINT @41,I;" REQUEST COMPLETE"
210 NEXT I
220 PRINT "SIGNING OFF"
230 ST$=CHR$(7)
240 PRINT #2,"DISC"
250 GOSUB 4000
260 CLOSE
270 CALL 21179
280 END
4000 Z$=INPUT$(1,1)
4010 IF Z$=ST$ THEN RETURN
4020 PRINT #3,Z$;
4030 GOTO 4000
5000 DATA 3
5010 DATA ",TAN", ",CIMN", "#BLHZ"
```

The following is a line-by-line description of the Auto Log-on program:

Program Line	Comment
10	Sets variable ST$ to CHR$(19) which is Control-S from ASCII character table. This character is the last one which Dow Jones sends at the end of each page of information
20	Sets PH$ to be equal to: **5551234** This is the phone number to call. < Forces unit to expect auto log-on. **?p** Wait for "p" from TYMNET. This is the first letter of "please type your terminal identifier. **A** You are terminal type A. **?P** Wait for "P" from TYMNET. This is the first letter in "Please log-in." **DOW1;;** This is your response. **?W** Wait for a "W" from Dow Jones. This is the first letter in "WHAT SERVICE PLEASE." **DJNS** You need to send DJNS. **^M** Send a Control-M; this is the same as pressing ENTER. **?P** Wait for a "W" from Dow Jones. This is the first letter in "PASSWORD." **password** This is your secret password. **^M** Send a Control-M; this is the same as pressing ENTER. > Terminates log-on sequences and causes the Model 100 to enter the Terminal Mode for interactive communications.
30	Set the variable M to be the address of the string variable PH$.
40	Set the variable AD to point to the beginning location in memory where the actual characters which make up PH$ are stored.
50	Execute a machine-language call which causes the built-in modem to lift the line to the telephone.
60	Execute a machine-language call which causes an auto-dial log-on sequence with the characters which begin in location pointed to by variable AD. Note that the middle character in the call is a 0 since the CPU register known as the accumulator is not used in this call routine.
70	Clears the Display.
80	Causes the built-in modem to be opened as an input device on channel #1 with the following communication parameters: 7 bit word length, Ignore parity, 1 stop bit, and Disable XON/XOFF.
90	Causes the built-in modem to be opened as an output device on channel #2.

100	Creates a Text file named QUOTE.DO to receive information from Dow Jones.
110	This retrieves one character from channel #1 and assigns it to variable Z$.
120	If Z$ is CHR$(19), then the first full page of sign-on information has been sent by Dow Jones.
130	Prints the current date and time to QUOTE.DO.
140	Prints on the Display that the information is being asked for.
150	Sets the variable N equal to the number of quote requests you will ask for.
160	Loop in lines 170-200 a number of times equal to the number of quotes you will request.
170	Sets Q$ equal to the first quote you want to see.
180	Sends Q$ out the modem line.
190	Go and get all the characters which will be coming back in answer to your request from Dow Jones.
200	Prints on the Display that the request is complete.
210	Process the next request.
220	Prints on the Display that you are signing-off.
230	Sets ST$ to be equal to the ASCII code for bell. This is the last character sent from Dow Jones.
240	Prints DISC to the modem.
250	Saves all the characters to the file. This will record the connect time.
260	Disconnects the modem as an I/O device and closes QUOTE.DO.
270	A machine-language call which will disconnect the phone line.
280	The end of the program.
4000	This retrieves one character from channel #1 and assigns it to Z$.
4010	Tests to see if the character received is the "Stop" character which signals the end of a page of information.
4020	Puts character in Z$ into QUOTE.DO that is active as channel #3. The trailing semi-colon causes the character to be saved without a trailing carriage return.
4030	Go back to line 4000.

5000	3 is the number of quotes that you will ask Dow Jones for. You should change this number for the number of quotes you want.
5010	Substitute the codes you want to obtain information about here. Be sure to add any necessary leading commas or other symbols as required by Dow Jones.

Sample Session #3: Sorting .DO files with a BASIC Program

The following BASIC program lets you manipulate and rearrange data stored in a .DO file. This includes alphabetically arranging lists of names in an ADRS.DO file. When using this program, it is imperative to keep a consistent format when creating the .DO file. For example, you can reserve the first 8 spaces in a line for first names, the next 10 spaces for last names, then 9 spaces for telephone numbers, and the remaining spaces for addresses.

Another practical application of the Sorter program is its ability to list a group of data in various formats. You can simply enter the information as soon as it is known and then let the program sort it for you — in any order that you want.

As you run the program, you'll be asked for the file you wish to sort. All existing files will appear on the Display. Answer by typing the name of the .DO file and pressing ENTER.

The Screen will change to show a line at the top with dashes, dots, and numbers, followed by a second line showing the first record of the .DO file you specified. You'll be asked to specify the position in the line to begin the sorting.

(The top line serves to indicate the column numbering. Number one shows the tenth column, number two shows the twentieth column, and so on. The dots are midpoints between these numbers.)

You should answer by referring to the line displayed at the top of the Screen in relation to the first line of the file you specified.

Having chosen a starting position by picking the proper column, you then will be asked for an ending position. Here you should type the column number to include all the characters that will be used by the sorter program.

When the program is through sorting according to code category, the OK prompt will appear on the Screen. You may then go to the Main Menu (by pressing F8), and examine .DO file.

On the other hand, if you want to examine the information not only according to categories, but also by date, simply specify the ending column to include the dates. You may use any column in the line to signal the beginning of the sorting process. This means you can even sort the file according to amount spent.

```
1000 ' This program sorts a data
1010 ' file stored in RAM. The file must
1020 ' be a data file, stored in ASCII
1030 ' format.  The program uses a
1040 ' Shell-Metzner sorting algorithm.
1050 '
1060 CLS
1070 CLEAR 2000
1080 FILES
1090 '
```

```
1100 ' Input the filename and verify
1110 ' it has a .DO extension
1120 '
1130 A$ = "Which file to sort: " : GOSUB 2000
1140 INPUT F$
1150  IF MID$(F$,LEN(F$)-2,1) <> "." THEN F$ = F$ + ".DO"
1170 OPEN F$ FOR INPUT AS 1
1180 '
1190 ' Print the first record of the
1200 ' file and determine the begin
1210 ' and end position of the sort
1220 ' field, and whether the field
1230 ' is numeric (F=1) or character
1240 ' (F=0)
1250 '
1260 LINE INPUT #1,Z$
1270 CLS
1280 PRINT"----.----1----.----2----.----3----.----4";
1290 PRINT Z$
1300 A$ = "Begin at position: " : GOSUB 2000
1310 INPUT B
1315 IF B=0 THEN1300
1320 A$ = "End at position: " : GOSUB 2000
1330 INPUT E
1370 N = 1
1380 '
1390 ' Input the remainder of the file
1400 ' to determine the size for the
1410 ' DIM statement.
1420 '
1430 N = N + 1
1440 LINE INPUT #1,Z$
1450 IF EOF(1) THEN GOTO 1470
1460 GOTO 1430
1470 CLOSE
1480 DIM D$(N)
1490 '
1500 ' Read in the data from the file
1510 '
1520 '
1530 '
1540 OPEN F$ FOR INPUT AS 1
1550 FOR I=1 TO N
1560 LINE INPUT #1,D$(I)
1570 NEXT I
1580 CLOSE 1
1600 '
1610 GOSUB 3000 'Call the sort routine
1620 '
1630 'Write the sorted file out to RAM
1640 '
1645 KILL F$
1650 OPEN F$ FOR OUTPUT AS 1
1660 FOR I=1 TO N
1670 PRINT #1,D$(I)
1680 NEXT I
1690 CLOSE
1700 '
1710 END:'CHANGE BACK TO MENU
2000 '
2010 ' Subroutine for printing prompts
2020 '
2030 PRINT @240, STRING$(40,32);
2040 PRINT @240, A$;
2050 RETURN
3000 '
3010 ' Sorting subroutine
3020 '
3030 Z5 = N
3040 Z5 = INT(Z5/2)
3050 IF Z5 = 0 THEN 3190
3060 Z2 = 1 : Z3 = N - Z5
3070 Z1 = Z2
3080 Z4 = Z1 + Z5
3100 IF (MID$(D$(Z1),B,(E-B)+1)) < (MID$(D$(Z4),B,(E-B)+1)) THEN 3160 ELSE 3120
3120 Z6$ = D$(Z1) : D$(Z1) = D$(Z4) : D$(Z4) = Z6$
```

```
3130 Z1 = Z1 - Z5
3140 IF Z1 < 1 THEN 3160
3150 GOTO 3080
3160 Z2 = Z2 + 1
3170 IF Z2 > Z3 THEN 3040
3180 GOTO 3070
3190 RETURN
```

Sample Session #4: ‘‘Fancy’’ Program Listings

This program sends a BASIC program listing to the printer in a ‘‘fancy’’ format. It paginates the listing to 56 lines per page, printing a title, as well as the time and date at the top of every page.

Multiple commands (separated by ‘‘:’’), will be printed in separate lines and will also be indented from the first command. Note that the file must be stored in an ASCII format (**SAVE** ‘‘*filename*’’**,A**).

```
1000 CLS: MAXFILES = 1
1010 CLEAR 2000
1020 PG = 0
1030 Z = 66
1040 DT$ = DATE$ : TM$ = TIME$
1050 FILES
1060 PRINT : INPUT "Name of Program:";N$
1070 OPEN N$ FOR INPUT AS 1
1080 PG = PG + 1 : LPRINT "***** Listing of Program '";N$;"' *****        ";DT$;"
- ";TM$;"....Page
1090 LPRINT
1100 LC = 2
1110 IN$ = INPUT$ (1,1):IF EOF(1) THEN Z = 132: GOTO 1230
1120 PR$ = PR$ + IN$
1130 IF IN$ <> ":" AND IN$ <> CHR$(10) THEN 1110
1140 IF IN$ = CHR$(10) THEN PR$ = LEFT$(PR$,LEN(PR$)-2)
1160 LPRINT " ": LC = LC +1: GOTO 1180
1170 LPRINT "      "
1180 LPRINT PR$
1190 PR$ = ""
1200 LC = LC + 1
1210 IF LC => 56 THEN 1230
1220 GOTO 1110
1230 FOR J = LC TO Z: LPRINT "": LC = LC + 1: NEXT J
1240 IF Z = 132 THEN 1260
1250 GOTO 1080
1260 END
```

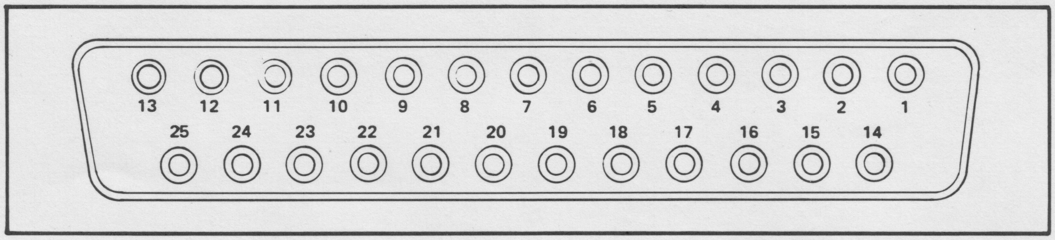
Appendix D / Technical Information

Power Source: AM Battery (x4) (23-552)
AC Adapter (DC 6V, Center minus) (26-3804)
Weight: 3 lbs. 13.5 oz. (1.3608 kg)
Dimensions: $11\frac{4}{5}$″ (L) x $8\frac{4}{9}$″ (D) x 2″ (H)
(30 cm x 21.5 cm x 5.08 cm)
Temperature: Operating 41°F (5°C) to 104°F (40°C)
Storage −40°F (−40° C) to 160°F (71°C)
Humidity: Operating 20% to 85% RH (non-condensing)
Storage 10% to 95% RN (non-condensing)

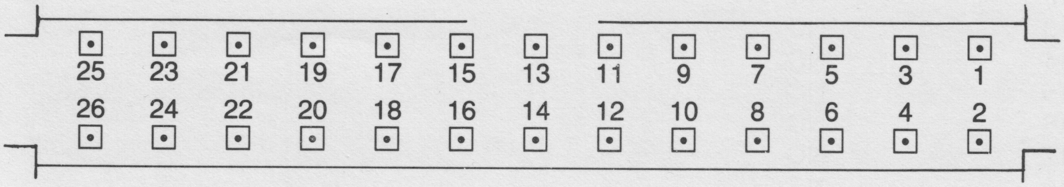
Micro Processor: 80C85 (8 bits CPU) 2.4 MHz

RS-232C Interface

Pin No.	Symbol	Description
1	GND	
2	TXR	Transmit Data
3	RXR	Receive Data
4	RTS	Request to send
5	CTS	Clear to send
6	DSR	Data set ready
7	GND	
8	NC	
9	NC	
10	NC	
11	NC	
12	NC	
13	NC	
14	NC	
15	NC	
16	NC	
17	NC	
18	NC	
19	NC	
20	DTR	Data terminal ready
21	NC	
22	NC	
23	NC	
24	NC	
25	NC	

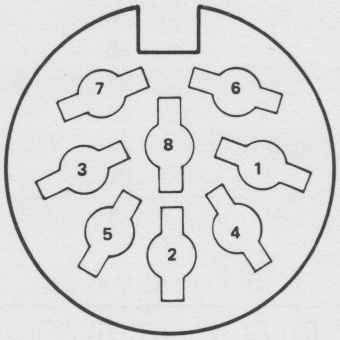


Parallel Printer Interface (Centronics)		
Pin No.	**Symbol**	**Description**
1	STROBE	Strobe pulse from the Computer to printer.
2	GND	Ground
3	PDO	Bit 0(1sb) of output data byte
4	GND	Ground
5	PD1	Bit 1 of output data byte
6	GND	Ground
7	PD2	Bit 2 of output data byte
8	GND	Ground
9	PD3	Bit 3 of output data byte
10	GND	Ground
11	PD4	Bit 4 of output data byte
12	GND	Ground
13	PD5	Bit 5 of output data byte
14	GND	Ground
15	PD6	Bit 6 of output data byte
16	GND	Ground
17	PD7	Bit 7 of output data byte
18	GND	Ground
19	NC	
20	GND	Ground
21	BUSY	Input to Computer from Printer
22	GND	Ground
23	NC	
24	GND	Ground
25	BUSY	Input to Computer from Printer, high indicates device selected.
26	NC	

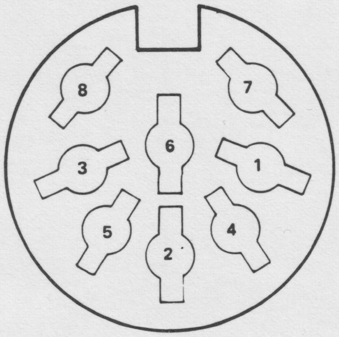


Cassette Interface		
Pin No.	Symbol	Description
1	REM 1	Remote
2	GND	
3	REM 2	Remote
4	R x C	Receive data for CMT
5	T x C	Transmit data for CMT
6	GND	GND
7	NC	
8	NC	

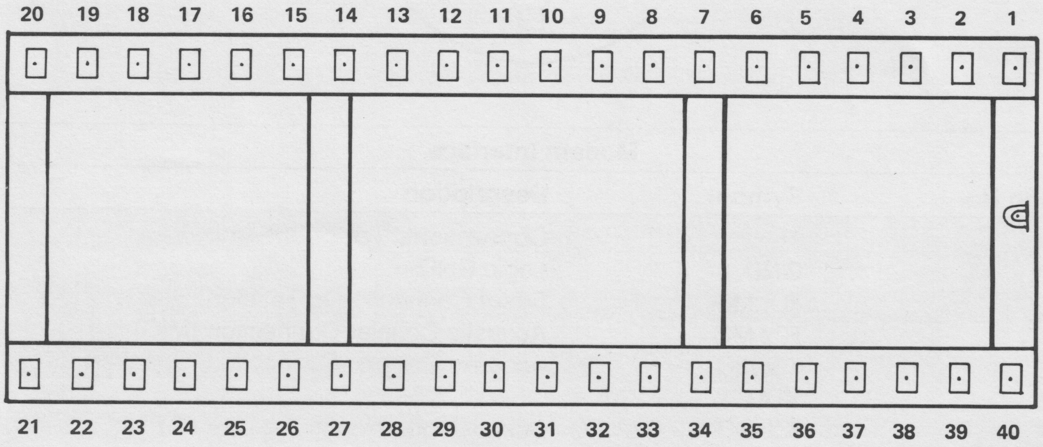
Input level: Inpedance 100ohm (800mV - 5Vpp)
Output level: Inpedance 3.3Kohm (650mVpp)
REMote: 6 VDC 0.5A max.



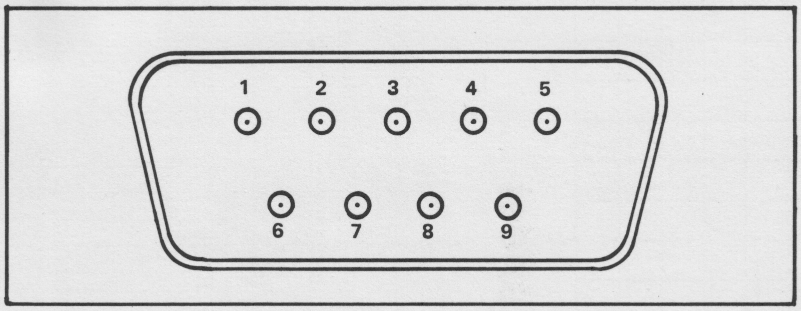
Modem Interface		
Pin No.	**Symbol**	**Description**
1	TL	Conventional Telephone Unit
2	GND	Logic GND
3	R x MD	Direct Connection to Tel Line (RING)
4	R x MC	Acoustic Coupler Connection (MIC)
5	T x MC	Acoustic Coupler Connection (Speaker)
6	VDD	
7	T x MD	Direct Connection to Tel Line (TIP)
8	RP	Ringing Paluse



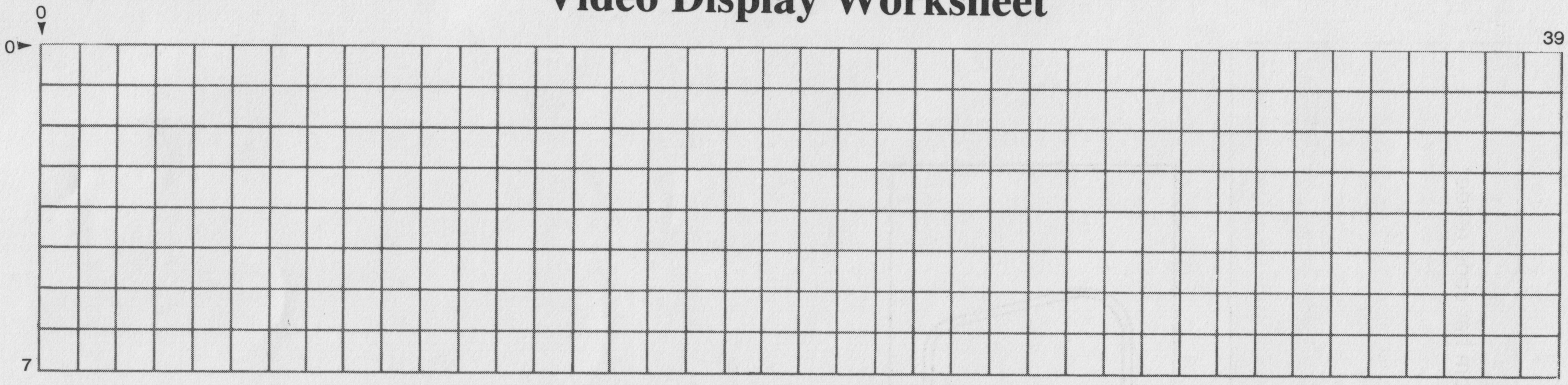
40 Pin External Bus Signal			
Pin No.	Signal	Pin No.	Signal
1	VDD	40	VDD
2	GND	39	GND
3	ADO	38	AD1
4	AD2	37	AD3
5	AD4	36	AD5
6	AD6	35	AD7
7	A8	34	A9
8	A10	33	A11
9	A12	32	A13
10	A14	31	A15
11	GND	30	GND
12	$\overline{RD}$	29	$\overline{WR}$
13	IO/M	28	S0
14	ALE	27	S1
15	CLK	26	CE for I/O Cont.
16	RD + WR	25	RESET out
17	$\overline{INTA}$	24	$\overline{INTA}$
18	GND	23	GND
19	RAM RAM RESET	22	NC
20	NC	21	NC


20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40

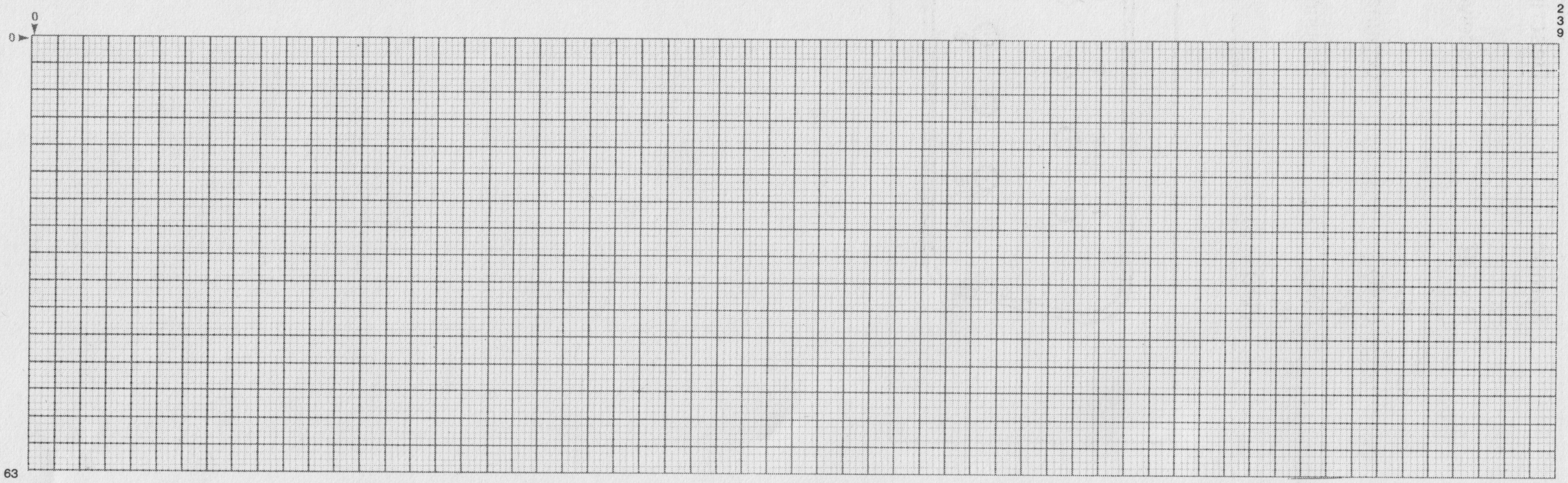
Bar Code Reader		
Pin No.	**Symbol**	**Description**
1	NC	
2	R x DB	Receive data from bar code reader
3	NC	
4	NC	
5	GND	Ground
6	NC	
7	GND	Ground
8	NC	
9	VDD	



Video Display Worksheet



Print @ Worksheet



Graphics Worksheet

ASCII Character Code Tables

Decimal	Hex	Binary	Printed Character	Keyboard Character
0	00	00000000		PAUSE
1	01	00000001		CTRL A
2	02	00000010		CTRL B
3	03	00000011		CTRL C
4	04	00000100		CTRL D
5	05	00000101		CTRL E
6	06	00000110		CTRL F
7	07	00000111		CTRL G
8	08	00001000		CTRL H
9	09	00001001		CTRL I
10	0A	00001010		CTRL J
11	0B	00001011		CTRL K
12	0C	00001100		CTRL L
13	0D	00001101		CTRL M
14	0E	00001110		CTRL N
15	0F	00001111		CTRL O
16	10	00010000		CTRL P
17	11	00010001		CTRL Q
18	12	00010010		CTRL R
19	13	00010011		CTRL S
20	14	00010100		CTRL T
21	15	00010101		CTRL U
22	16	00010110		CTRL V
23	17	00010111		CTRL W
24	18	00011000		CTRL X
25	19	00011001		CTRL Y
26	1A	00011010		CTRL Z
27	1B	00011011		ESC
28	1C	00011100		→
29	1D	00011101		←
30	1E	00011110		↑
31	1F	00011111		↓
32	20	00100000		SPACEBAR
33	21	00100001	!	!
34	22	00100010	"	"
35	23	00100011	#	#
36	24	00100100	$	$
37	25	00100101	%	%
38	26	00100110	&	&
39	27	00100111	'	'
40	28	00101000	(	(
41	29	00101001	)	)
42	2A	00101010	*	*
43	2B	00101011	+	+
44	2C	00101100	,	,
45	2D	00101101	–	–
46	2E	00101110	.	.

Decimal	Hex	Binary	Printed Character	Keyboard Character
47	2F	00101111	/	/
48	30	00110000	0	0
49	31	00110001	1	1
50	32	00110010	2	2
51	33	00110011	3	3
52	34	00110100	4	4
53	35	00110101	5	5
54	36	00110110	6	6
55	37	00110111	7	7
56	38	00111000	8	8
57	39	00111001	9	9
58	3A	00111010	:	:
59	3B	00111011	;	;
60	3C	00111100	<	<
61	3D	00111101	=	=
62	3E	00111110	>	>
63	3F	00111111	?	?
64	40	01000000	@	@
65	41	01000001	A	A
66	42	01000010	B	B
67	43	01000011	C	C
68	44	01000100	D	D
69	45	01000101	E	E
70	46	01000110	F	F
71	47	01000111	G	G
72	48	01001000	H	H
73	49	01001001	I	I
74	4A	01001010	J	J
75	4B	01001011	K	K
76	4C	01001100	L	L
77	4D	01001101	M	M
78	4E	01001110	N	N
79	4F	01001111	O	O
80	50	01010000	P	P
81	51	01010001	Q	Q
82	52	01010010	R	R
83	53	01010011	S	S
84	54	01010100	T	T
85	55	01010101	U	U
86	56	01010110	V	V
87	57	01010111	W	W
88	58	01011000	X	X
89	59	01011001	Y	Y
90	5A	01011010	Z	Z
91	5B	01011011	[	[
92	5C	01011100	\	GRPH –
93	5D	01011101	]	]
94	5E	01011110	^	^

* For uppercase letters A-Z, press SHIFT or CAPS LOCK before pressing the Keyboard Character.

Appendices

Decimal	Hex	Binary	Printed Character	Keyboard Character
95	5F	01011111	_	_
96	60	01100000	\	GRAPH [
97	61	01100001	a	A
98	62	01100010	b	B
99	63	01100011	c	C
100	64	01100100	d	D
101	65	01100101	e	E
102	66	01100110	f	F
103	67	01100111	g	G
104	68	01101000	h	H
105	69	01101001	i	I
106	6A	01101010	j	J
107	6B	01101011	k	K
108	6C	01101100	l	L
109	6D	01101101	m	M
110	6E	01101110	n	N
111	6F	01101111	o	O
112	70	01110000	p	P
113	71	01110001	q	Q
114	72	01110010	r	R
115	73	01110011	s	S
116	74	01110100	t	T
117	75	01110101	u	U
118	76	01110110	v	V
119	77	01110111	w	W
120	78	01111000	x	X
121	79	01111001	y	Y
122	7A	01111010	z	Z
123	7B	01111011	{	GRPH 9
124	7C	01111100	|	GRPH _
125	7D	01111101	}	GRPH 0
126	7E	01111110	~	GRPH]
127	7F	01111111		DEL
128	80	10000000	☎	GRPH p
129	81	10000001	🏠	GRPH m
130	82	10000010	(×	GRPH f
131	83	10000011	◪	GRPH x
132	84	10000100	🚗	GRPH c
133	85	10000101	✈	GRPH a
134	86	10000110	🏢	GRPH h
135	87	10000111	🏣	GRPH t
136	88	10001000	i	GRPH l
137	89	10001001	√	GRPH r
138	8A	10001010	≠	GRPH /
139	8B	10001011	Σ	GRPH s
140	8C	10001100	≈	GRPH '
141	8D	10001101	±	GRPH =
142	BE	10001110	∫	GRPH i
143	BF	10001111	◀	GRPH e

* For lowercase letters a-z, be sure CAPS LOCK is not pressed "down."

Decimal	Hex	Binary	Printed Character	Keyboard Character
144	90	10010000	Π	GRPH y
145	91	10010001	∩	GRPH u
146	92	10010010	↕	GRPH ;
147	93	10010011	🯅	GRPH q
148	94	10010100	🯆	GRPH w
149	95	10010101	♂	GRPH b
150	96	10010110	♀	GRPH n
151	97	10010111	℅	GRPH .
152	98	10011000	↑	GRPH o
153	99	10011001	↓	GRPH ,
154	9A	10011010	→	GRPH l
155	9B	10011011	←	GRPH k
156	9C	10011100	♧	GRPH 2
157	9D	10011101	♢	GRPH 3
158	9E	10011110	♡	GRPH 4
159	9F	10011111	♤	GRPH 5
160	A0	10100000	´	CODE '
161	A1	10100001	à	CODE x
162	A2	10100010	ç	CODE c
163	A3	10100011	£	GRPH 8
164	A4	10100100	`	CODE "
165	A5	10100101	µ	CODE M
166	A6	10100110	°	CODE)
167	A7	10100111	▼	CODE _
168	A8	10101000	†	CODE +
169	A9	10101001	§	CODE s
170	AA	10101010	🆁	CODE R
171	AB	10101011	🅲	CODE C
172	AC	10101100	¼	CODE p
173	AD	10101101	¾	CODE ;
174	AE	10101110	½	CODE /
175	AF	10101111	¶	CODE 0
176	B0	10110000	¥	GRPH 7
177	B1	10110001	Ä	CODE A
178	B2	10110010	Ö	CODE O
179	B3	10110011	Ü	CODE U
180	B4	10110100	¢	GRPH 6
181	B5	10110101	~	CODE [
182	B6	10110110	ä	CODE a
183	B7	10110111	ö	CODE o
184	B8	10111000	ü	CODE u
185	B9	10111001	B	CODE S
186	BA	10111010	™	CODE T
187	BB	10111011	é	CODE d
188	BC	10111100	ù	CODE ,
189	BD	10111101	è	CODE v
190	BE	10111110	¨	CODE =
191	BF	10111111	£	CODE F
192	C0	11000000	â	CODE l
193	CI	11000001	ê	CODE 3

Decimal	Hex	Binary	Printed Character	Keyboard Character
194	C2	11000010	î	CODE 8
195	C3	11000011	ô	CODE 9
196	C4	11000100	û	CODE 7
197	C5	11000101	ˆ	CODE –
198	C6	11000110	ë	CODE e
199	C7	11000111	ï	CODE i
200	C8	11001000	á	CODE q
201	C9	11001001	í	CODE k
202	CA	11001010	ó	CODE l
203	CB	11001011	ú	CODE j
204	CC	11001100	ý	CODE y
205	CD	11001101	ñ	CODE n
206	CE	11001110	ã	CODE z
207	CF	11001111	õ	CODE .
208	D0	11010000	Â	CODE !
209	D1	11010001	Ê	CODE #
210	D2	11010010	Î	CODE *
211	D3	11010011	Ô	CODE (
212	D4	11010100	Û	CODE &
213	D5	11010101	Ï	CODE I
214	D6	11010110	Ë	CODE E
215	D7	11010111	É	CODE D
216	D8	11011000	Á	CODE Q
217	D9	11011001	Í	CODE K
218	DA	11011010	Ó	CODE L
219	DB	11011011	Ú	CODE J
220	DC	11011100	Ý	CODE Y
221	DD	11011101	Ù	CODE <
222	DE	11011110	È	CODE V
223	DF	11011111	À	CODE X
224	ED	11100000		GRPH Z
225	E1	11100001	▘ (upper left)	GRPH !
226	E2	11100010	▝ (upper right)	GRPH @
227	E3	11100011	▖ (lower left)	GRPH #
228	E4	11100100	▗ (lower right)	GRPH $
229	E5	11100101	▚	GRPH %
230	E6	11100110	▞	GRPH
231	E7	11100111	▔ (upper)	GRPH Q
232	E8	11101000	▁ (lower)	GRPH W
233	E9	11101001	▌ (left)	GRPH E
234	EA	11101010	▐ (right)	GRPH R
235	EB	11101011	▛	GRPH A
236	EC	11101100	▜	GRPH S
237	ED	11101101	▙	GRPH D
238	EE	11101110	▟	GRPH F
239	EF	11101111	█	GRPH X
240	F0	11110000	┌	GRPH U
241	F1	11110001	─	GRPH P
242	F2	11110010	┐	GRPH O
243	F3	11110011	┬	GRPH I

Decimal	Hex	Binary	Printed Character	Keyboard Character
244	F4	11110100	├	GRPH J
245	F5	11110101	│	GRPH :
246	F6	11110110	└	GRPH M
247	F7	11110111	┘	GRPH >
248	F8	11111000	┴	GRPH <
249	F9	11111001	┤	GRPH L
250	FA	11111010	┼	GRPH K
251	FB	11111011	◤	GRPH H
252	FC	11111100	◢	GRPH T
253	FD	11111101	◥	GRPH G
254	FE	11111110	◣	GRPH Y
255	FF	11111111	▒	GRPH C

BASIC Error Codes

Code	Message	Meaning
1	NF	NEXT without FOR.
2	SN	Syntax Error.
3	RG	RETURN without GOSUB.
4	OD	Out of Data.
5	FC	Illegal function call.
6	OV	Overflow.
7	OM	Out of Memory.
8	UL	Undefined line.
9	BS	Bad Subscript.
10	DD	Doubly Dimensioned Array.
11	/0	Division by Zero.
12	ID	Illegal Direct.
13	TM	Type Mismatch.
14	OS	Out of String Space.
15	LS	String Too Long.
16	ST	String Formula Too Complex.
17	CN	Can't Continue.
18	IO	Error.
19	NR	No RESUME.
20	RW	RESUME Without Error.
21	UE	Undefined Error.
22	MO	Missing Operand.
23-49	UE	Undefined Error.
50	IE	Undefined Error.
51	BN	Bad File Number.
52	FF	File Not Found.
53	AO	Already Open.
54	EF	Input Past End of File.
55	NM	Bad file name.
56	DS	Direct Statement in File.
57	FL	Undefined error.
58	CF	File Not Open.
59-255	UE	Undefined Error.

Derived Functions

Function	Function Expresses in Terms of Model 100 BASIC Functions. X is in radians.
SECANT	SEC(X) = 1/COS(X)
COSECANT	CSC(X) = 1/SIN(X)
COTANGENT	COT(X) = 1/TAN(X)
INVERSE SINE	ARCSIN(C) = ATN(X/SQR(-X*X+1))
INVERSE COSINE	ARCCOS(X) = -ATN(X/SQR(-X*X+1))+1.5708
INVERSE SECANT	ARCSEC(X) = ATN (SQR(X*X-1)) +(SGN(X)-1)*1.5708
INVERSE COSECANT	ARCCSC(X) = ATN(1/SQR(X*X-1)) +(SGN(X)-1)*1.5708
INVERSE COTANGENT	ARCCOT(X) = -ATN(X)+1.5708
HYPERBOLIC SINE	SINH(X) = (EXP(X)-EXP(-X))/2
HYPERBOLIC COSINE	COSH(X) = (EXP(X)+EXP(-X))/2
HYPERBOLIC TANGENT	TANH(X) = -EXP(-X)/(EXP(X)+EXP(-X))*2+1
HYPERBOLIC SECANT	SECH(X) = 2/(EXP(X)+EXP(-X))
HYPERBOLIC COSECANT	CSCH(X) = 2/(EXP(X)-EXP(-X))
HYPERBOLIC COTANGENT	COTH(X) = EXP(-X)/(EXP(X)-EXP(-X))*2+1
INVERSE HYPERBOLIC SINE	ARCSINH(X) = LOG(X+SQR(X*X+1))
INVERSE HYPERBOLIC COSINE	ARCCOSH(X) = LOG(X+SQR(X*X-1))
INVERSE HYPERBOLIC TANGENT	ARCTANH(X) = LOG((1+X)/(1-X))/2
INVERSE HYPERBOLIC SECANT	ARCSECH(X) = LOG((SQR(-X*X+1)+1)/X)
INVERSE HYPERBOLIC COSECANT	ARCCSCH(X) = LOG((SGN(X)*SQR(X*X+1)+1)/X)
INVERSE HYPERBOLIC COTANGENT	ARCCOTH(X) = LOG((X + 1)/(X-1)/2
PI	4 * ATN(1)

	Valid Input Range
Inverse Sine	-1 < X < 1
Inverse Cosine	-1 < X < 1
Inverse Secant	X < -1 OR X > 1
Inverse Cosecant	X < -1 OR X > 1
Inverse Hyper, Cosine	X > 1
Inverse Hyper, Tangent	X*X < 1
Inverse Hyper, Secant	0 < X < 1
Inverse Hyper, Cosecant	X < > 0
Inverse Hyper, Cotangent	X*X > 1

Appendix E / Troubleshooting and Maintenance

If you have problems operating your Model-100, read the instructions again, paying attention in particular to the sequential order of steps.

If you continue having trouble, check the following table to identify the symptoms and possible cures.

If you still cannot solve the problems after doing this, take the unit to the nearest Radio Shack Repair Center. We'll have it fixed and returned to you as soon as possible.

Symptom	**Possible Cause/Cure**
The Menu message does not appear when you turn on the Computer.	1. Check whether the AC adapter is correctly connected to power plug. 2. Check whether the AC adapter plug is correctly connected to the DC jack on the side of the Computer. 3. Check whether the Memory Power Switch is ON. 4. Check whether the Power-On Switch is ON. 5. Adjust the DISP switch to check brightness and contrast controls. 6. If using AM batteries, check whether batteries are inserted properly or if the Low-Power Sensor LED light is on. (If it is on, replace the batteries immediately.)

If all the above cures do not work, press the CTRL and PAUSE simultaneously and turn on the Power Switch.

Maintenance

Your Computer requires little maintenance. However, it is a good idea to keep it clean and free of dust build-up. This is especially important for smooth keyboard operation.

Store the Computer inside the case when not in use.

Index

Subject	Page
AC Power Connection	11
Acoustic Coupler	8,75,76,78
ADDRSS	
Applications	69
Command Keys	27,70
Function Keys	26,69
Quick Instructions	26
ADRS.DO	
Creating a File	27,70
Retrieving Information	27,71
ANSwer/ORIGinate Selector	8,85
Arctangent	127
Arrays	105
Assignment Statements	106,112
ATN	127
Auto-dialing	31,83
Auto-Power Off	195
Auto Log-on Sequence	32,89,91,199,200
Guidelines	90,91
BASIC	
Error Codes	217
Operation Modes	99
Overview	99
Starting Up	99
BEEP	123,127
BKSP DEL	7
Bar Code Reader	209
Bar Code Wand Connector	8
Battery	
Compartment	8
Installation	9
CALL	117,127
CAPS LOCK	7
Cassette Files	123,124
Cassette Interface	207
CASSETTE Recorder Connector	8,189
CDBL	128
CHR$	128
CINT	128
CLEAR	129
CLOAD	123,129
CLOAD?	123,130
CLOADM	123,130
CLOSE	131
CLS	100,122,131
CODE	7
COM ON/OFF/STOP	118,131
Command Keys	6,41
Definitions	20,24,27,31,41,44,62,70
Command Mode	99
Special Keys	100
Communication Parameters	34,85
Change	87
Verify	86
Computer-to-Computer Communication(TELCOM)	75
Constants	104
CONT	100,132
Control Code Combinations	59,60
Control Commands	115
COPY	49,56
COS	132
CPU Ports	123
CSAVE	132
CSAVEM	133
CSNG	133
CSRLIN	118,134
CTRL	7
Cursor	15
Cursor Movement Keys	6,15,41,45
CUT	49,53,55
DATA	134
Data, Representing	104
Data, String	104
Data Conversion	113
Data Manipulation	103
Data Types	103
DATE$	134
Date, Setting	18
DAY$	18, 135
Day, Setting	18
DEFDBL	106,135
DEFINT	106,135
DEFSNG	106,136
DEFSTR	106,136
DIM	105,136
Derived Functions	218
Direct Connect/Acoustic Coupler Modem Selector	8
Direct Connect Modem (PHONE) Connector	8
Display Adjustment Dial	5,16
Double-Precision Numbers	103,114
Double-Precision to Single-Precision	114
Download	30,93,94
Downloading Files	198
EDIT	137
Edit Mode	101
END	138
ENTER	7
Entry Mode	28,29,31,75
Entry Mode, Using	31,82
EOF	138
ERL	139

Index

Subject **Page**

ERR 139
ERROR 140
ESC 6
Execute Mode 101
  Special Keys 101
EXP 140
Expressions 106-111
External Bus Signal 208
External Power Adapter Connector 5

File NOTE.DO 62
File Names 40,45
File I/O 121,124,126,165
File Deletion 40
FIND Command 52
FIND Function 57
FIX 141
FOR...NEXT 112,116,141
FRE 100,142
Function Keys 6
  Definitions 19,23,26,29,30,41,43,61,69
  Entry Mode 29,79
  Terminal Mode 30,81
  Usage 40,41

GOSUB 116,143
GOTO 115, 143
GRPH 7

HIMEM 100,144

ID Tag 8
IF...THEN...ELSE 111,116,144
Immediate I/O 121
Immediate Line 100
Integer Numbers 103,114
Integer to Single- or Double-Precision 114
Interrupt Commands 117,118
INKEY$ 145
INP 145
INPUT 118,141,145
INPUT# 146
INPUT$ 147
INSTR 147
INT 148
IPL 110,148

KEY 149
KEY LIST 149
KEY ON/OFF/STOP 149
KILL 100,150
Keyboard 5
Keyboard Input 122

Subject **Page**

LCOPY 100,150
LEFT$ 151
LEN 151
LET 151
LINE 152
LINE INPUT 153
LIST 100,152
LLIST 100,153
LOAD 100,154
LOAD Function 58
LOADM 155
LOG 155
Logical Expressions 107,111
Log-On 32
  Automatic 32,89,90
Log-On Sequence 32,33
Low Battery Indicator 6,10
LPOS 156
LPRINT 156
LPRINT USING 157
MAXFILES 157
MAXRAM 100,157
MDM ON/OFF/STOP 157
Memory Power Switch 8,10
MENU 100,158
Menu
  Overview 39
  Screen 15
  Selection 16
MERGE 100,158
MID$ 159,160
Modem
  Cable 75,76
  Files(MDM) 125
  Interface 207
MOTOR 123,160

NAME...AS 160
NEW 100,161
NOTE.DO File 23, 61
  Creating File 24,62
  Retrieving Information 24,63
  Scan 25,65
Null Modem Adapter 75,193
NUM 7, 17
Numeric Data 103
Numeric Expressions 107,110
Numeric Functions 108,109

ON COM GOSUB 161
ON ERROR GOTO 162
ON KEY GOSUB 163
ON MDM GOSUB 163
ON TIME$ GOSUB 163

Index

Subject Page
ON...GOSUB 164
ON...GOTO 164
OPEN 165
Operands 106
Operators 106
OUT 166

Parallel Printer Interface 206
PASTE 55,56
PASTE Buffer 53,55,56
PEEK 166
POKE 167
POS 167
POWER 100,167
    CONT 167
    OFF 168
Power
    ON/OFF Switch 5
    ON and OFF 13
PRESET 168
PRINT 168
PRINT# 169
PRINT # USING 172
PRINT USING 170
Printer
    Files(LPT) 125,126
    I/O 122
PRINTER Connector 7,192
Program Line 100
Program Name 40
Programmable Function Keys 6
PSET 172

RAM
    8K 1,17
    16K 1,17
    24K 1,17
    32K 1,17
RAM Files 124,160
READ 173
Relational Expressions 110
REM 173
RESET Button 7
RESTORE 174
RESUME 168,174
RIGHT$ 175
RND 175
ROM Module Expansion Compartment 8
RS-232C Files(COM) 125
RS-232C Connector 7
RUN 175,176
RUNM 177

SAVE 100,177

Subject Page
SAVE Function 59
SAVEM 178
SCHEDL
    Applications 61
    Command Keys 24,62
    Function Keys 23,61
    More and Quit Prompts 64,65,67
    Quick Instructions 23
    Sample Session 65
SCREEN 179
Screen 5,121
    Files(LCD) 125,126
    I/O 121,122
SGN 179
SHIFT 7
SIN 179
Single-Precision Numbers 103,114
Single- or Double-Precision to Integer 114
SOUND 180
Sound Generator 123
SOUND ON/OFF 123,180
SPACE$ 181
SQR 181
STOP 119,181
STR$ 182
STRING$ 182
String Data 104
String Expressions 109,110
String Functions 110

TAB 7,182
TAN 183
TELCOM 75
    Access 82
    Applications 75
    Automatic Entry (Terminal Mode) 32,75,88
    Command Keys 30,31
    Communication Parameters 85-87
    Entry Mode 75,79,82
    Exiting Terminal Mode 96
    Function Keys 29,30,79,81
    Manual Entry (Terminal Mode) 32,89
    Quick Instructions 28
    Telephone Connections 76
    Terminal Mode 29,32,75,81,84
Text Preparation (TEXT) 19,43
    Addition 48
    Applications 19
    Closing Text Files 20,46,47
    Command Keys 20,44
    Creating a Text File 20,45
    Cursor Movement Keys 45
    Define 22,49-52

Index

Subject **Page**

Delete....................21,22,40
Deleting Text..............21,53,54
Duplicating Text 23,49,52,56
Editing Text 21,47
Editing Functions 21,47
Function Keys 19,23,26,29,30,43
Insert 21,47,48
Manipulating Text Block..........22
Moving Text.................23,55
Quick Instructions 19
Opening Text Files 20
Printing Text Files 21,44

Subject **Page**

TIME$ 18,183
TIME$/ON/OFF/STOP 183
Time, Date, and Day, Setting 17, 18
Type Declaration Tags............105,106

Uploading Files 198

VAL.............................184
VARPTR 185
Variables 104,105,112

Warm vs. Cold Power On 195

SERVICE POLICY

Radio Shack's nationwide network of service facilities provides quick, convenient, and reliable repair services for all of its computer products, in most instances. Warranty service will be performed in accordance with Radio Shack's Limited Warranty. Non-warranty service will be provided at reasonable parts and labor costs.

Because of the sensitivity of computer equipment, and the problems which can result from improper servicing, the following limitations also apply to the services offered by Radio Shack:

1. If any of the warranty seals on any Radio Shack computer products are broken, Radio Shack reserves the right to refuse to service the equipment or to void any remaining warranty on the equipment.
2. If any Radio Shack computer equipment has been modified so that it is not within manufacturer's specifications, including, but not limited to, the installation of any non-Radio Shack parts, components, or replacement boards, then Radio Shack reserves the right to refuse to service the equipment, void any remaining warranty, remove and replace any non-Radio Shack part found in the equipment, and perform whatever modifications are necessary to return the equipment to original factory manufacturer's specifications.
3. The cost for the labor and parts required to return the Radio Shack computer equipment to original manufacturer's specifications will be charged to the customer in addition to the normal repair charge.

RADIO SHACK, A DIVISION OF TANDY CORPORATION

U.S.A.: FORT WORTH, TEXAS 76102
CANADA: BARRIE, ONTARIO L4M 4W5

TANDY CORPORATION

AUSTRALIA	BELGIUM	U. K.
91 KURRAJONG ROAD MOUNT DRUITT, N.S.W. 2770	PARC INDUSTRIEL DE NANINNE 5140 NANINNE	BILSTON ROAD WEDNESBURY WEST MIDLANDS WS10 7JN

2A3

PRINTED IN JAPAN